

activemq-cpp-3.2.1

Generated by Doxygen 1.7.1

Sun Sep 11 2011 18:23:35

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Data Structure Index	3
2.1	Class Hierarchy	3
3	Data Structure Index	25
3.1	Data Structures	25
4	File Index	61
4.1	File List	61
5	Namespace Documentation	83
5.1	activemq Namespace Reference	83
5.1.1	Detailed Description	83
5.2	activemq::cmsutil Namespace Reference	84
5.3	activemq::commands Namespace Reference	85
5.4	activemq::core Namespace Reference	86
5.5	activemq::core::policies Namespace Reference	87
5.6	activemq::exceptions Namespace Reference	87
5.7	activemq::io Namespace Reference	87
5.8	activemq::library Namespace Reference	88
5.9	activemq::state Namespace Reference	88
5.10	activemq::threads Namespace Reference	88
5.11	activemq::transport Namespace Reference	89
5.12	activemq::transport::correlator Namespace Reference	89
5.13	activemq::transport::failover Namespace Reference	90
5.14	activemq::transport::inactivity Namespace Reference	90
5.15	activemq::transport::logging Namespace Reference	90
5.16	activemq::transport::mock Namespace Reference	90

5.17	activemq::transport::tcp Namespace Reference	91
5.18	activemq::util Namespace Reference	91
5.19	activemq::wireformat Namespace Reference	92
5.20	activemq::wireformat::openwire Namespace Reference	92
5.21	activemq::wireformat::openwire::marshal Namespace Reference	93
5.22	activemq::wireformat::openwire::marshal::v1 Namespace Reference	93
5.23	activemq::wireformat::openwire::marshal::v2 Namespace Reference	97
5.24	activemq::wireformat::openwire::marshal::v3 Namespace Reference	101
5.25	activemq::wireformat::openwire::marshal::v4 Namespace Reference	105
5.26	activemq::wireformat::openwire::marshal::v5 Namespace Reference	109
5.27	activemq::wireformat::openwire::marshal::v6 Namespace Reference	113
5.28	activemq::wireformat::openwire::utils Namespace Reference	117
5.29	activemq::wireformat::stomp Namespace Reference	117
5.30	cms Namespace Reference	117
5.30.1	Detailed Description	120
5.31	decaf Namespace Reference	120
5.31.1	Detailed Description	121
5.32	decaf::internal Namespace Reference	121
5.33	decaf::internal::io Namespace Reference	121
5.34	decaf::internal::net Namespace Reference	122
5.35	decaf::internal::net::ssl Namespace Reference	122
5.36	decaf::internal::net::ssl::openssl Namespace Reference	123
5.37	decaf::internal::net::tcp Namespace Reference	123
5.38	decaf::internal::nio Namespace Reference	124
5.39	decaf::internal::security Namespace Reference	124
5.40	decaf::internal::util Namespace Reference	124
5.41	decaf::internal::util::concurrent Namespace Reference	125
5.42	decaf::io Namespace Reference	125
5.43	decaf::lang Namespace Reference	127
5.43.1	Function Documentation	129
5.43.1.1	operator!=	129
5.43.1.2	operator!=	129
5.43.1.3	operator!=	129
5.43.1.4	operator!=	129
5.43.1.5	operator==	129
5.43.1.6	operator==	130

5.43.1.7	operator==	130
5.43.1.8	operator==	130
5.44	decaf::lang::exceptions Namespace Reference	130
5.45	decaf::net Namespace Reference	130
5.46	decaf::net::ssl Namespace Reference	132
5.47	decaf::nio Namespace Reference	132
5.48	decaf::security Namespace Reference	133
5.49	decaf::security::auth Namespace Reference	133
5.50	decaf::security::auth::x500 Namespace Reference	133
5.51	decaf::security::cert Namespace Reference	134
5.52	decaf::util Namespace Reference	134
5.53	decaf::util::comparators Namespace Reference	136
5.54	decaf::util::concurrent Namespace Reference	136
5.55	decaf::util::concurrent::atomic Namespace Reference	138
5.56	decaf::util::concurrent::locks Namespace Reference	138
5.57	decaf::util::logging Namespace Reference	139
5.57.1	Enumeration Type Documentation	140
5.57.1.1	Levels	140
5.58	decaf::util::zip Namespace Reference	140
5.59	std Namespace Reference	141
6	Data Structure Documentation	143
6.1	decaf::util::AbstractCollection< E > Class Template Reference	143
6.1.1	Detailed Description	145
6.1.2	Constructor & Destructor Documentation	146
6.1.2.1	AbstractCollection	146
6.1.2.2	~AbstractCollection	146
6.1.3	Member Function Documentation	146
6.1.3.1	add	146
6.1.3.2	addAll	147
6.1.3.3	clear	147
6.1.3.4	contains	148
6.1.3.5	containsAll	149
6.1.3.6	copy	149
6.1.3.7	equals	149
6.1.3.8	isEmpty	150
6.1.3.9	lock	150

6.1.3.10	notify	150
6.1.3.11	notifyAll	151
6.1.3.12	operator=	151
6.1.3.13	remove	151
6.1.3.14	removeAll	152
6.1.3.15	retainAll	153
6.1.3.16	toArray	153
6.1.3.17	tryLock	154
6.1.3.18	unlock	154
6.1.3.19	wait	154
6.1.3.20	wait	154
6.1.3.21	wait	155
6.1.4	Field Documentation	155
6.1.4.1	mutex	155
6.2	decaf::util::AbstractList< E > Class Template Reference	156
6.2.1	Detailed Description	156
6.2.2	Constructor & Destructor Documentation	157
6.2.2.1	~AbstractList	157
6.3	decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference	157
6.3.1	Detailed Description	157
6.3.2	Constructor & Destructor Documentation	158
6.3.2.1	~AbstractMap	158
6.4	decaf::util::AbstractQueue< E > Class Template Reference	158
6.4.1	Detailed Description	159
6.4.2	Constructor & Destructor Documentation	159
6.4.2.1	AbstractQueue	159
6.4.2.2	~AbstractQueue	159
6.4.3	Member Function Documentation	159
6.4.3.1	add	159
6.4.3.2	addAll	160
6.4.3.3	clear	160
6.4.3.4	element	160
6.4.3.5	remove	161
6.5	decaf::util::AbstractSequentialList< E > Class Template Reference	161
6.5.1	Detailed Description	162
6.5.2	Constructor & Destructor Documentation	162

6.5.2.1	<code>~AbstractSequentialList</code>	162
6.6	<code>decaf::util::AbstractSet< E ></code> Class Template Reference	162
6.6.1	Detailed Description	163
6.6.2	Constructor & Destructor Documentation	163
6.6.2.1	<code>~AbstractSet</code>	163
6.6.3	Member Function Documentation	163
6.6.3.1	<code>removeAll</code>	163
6.7	<code>activemq::transport::AbstractTransportFactory</code> Class Reference	164
6.7.1	Detailed Description	164
6.7.2	Constructor & Destructor Documentation	165
6.7.2.1	<code>~AbstractTransportFactory</code>	165
6.7.3	Member Function Documentation	165
6.7.3.1	<code>createWireFormat</code>	165
6.8	<code>activemq::core::ActiveMQAckHandler</code> Class Reference	165
6.8.1	Detailed Description	165
6.8.2	Constructor & Destructor Documentation	166
6.8.2.1	<code>~ActiveMQAckHandler</code>	166
6.8.3	Member Function Documentation	166
6.8.3.1	<code>acknowledgeMessage</code>	166
6.9	<code>activemq::commands::ActiveMQBlobMessage</code> Class Reference	166
6.9.1	Constructor & Destructor Documentation	167
6.9.1.1	<code>ActiveMQBlobMessage</code>	167
6.9.1.2	<code>~ActiveMQBlobMessage</code>	167
6.9.2	Member Function Documentation	167
6.9.2.1	<code>clone</code>	167
6.9.2.2	<code>cloneDataStructure</code>	168
6.9.2.3	<code>copyDataStructure</code>	168
6.9.2.4	<code>equals</code>	168
6.9.2.5	<code>getDataStructureType</code>	168
6.9.2.6	<code>getMimeType</code>	169
6.9.2.7	<code>getName</code>	169
6.9.2.8	<code>getRemoteBlobUrl</code>	169
6.9.2.9	<code>isDeletedByBroker</code>	169
6.9.2.10	<code>setDeletedByBroker</code>	169
6.9.2.11	<code>setMimeType</code>	170
6.9.2.12	<code>setName</code>	170

6.9.2.13	setRemoteBlobUrl	170
6.9.2.14	toString	170
6.9.3	Field Documentation	170
6.9.3.1	BINARY_MIME_TYPE	170
6.9.3.2	ID_ACTIVEMQBLOBMESSAGE	170
6.10	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller	
	Class Reference	171
6.10.1	Detailed Description	172
6.10.2	Constructor & Destructor Documentation	172
6.10.2.1	ActiveMQBlobMessageMarshaller	172
6.10.2.2	~ActiveMQBlobMessageMarshaller	172
6.10.3	Member Function Documentation	172
6.10.3.1	createObject	172
6.10.3.2	getDataStructureType	172
6.10.3.3	looseMarshal	172
6.10.3.4	looseUnmarshal	173
6.10.3.5	tightMarshal1	173
6.10.3.6	tightMarshal2	174
6.10.3.7	tightUnmarshal	174
6.11	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	
	Class Reference	174
6.11.1	Detailed Description	175
6.11.2	Constructor & Destructor Documentation	176
6.11.2.1	ActiveMQBlobMessageMarshaller	176
6.11.2.2	~ActiveMQBlobMessageMarshaller	176
6.11.3	Member Function Documentation	176
6.11.3.1	createObject	176
6.11.3.2	getDataStructureType	176
6.11.3.3	looseMarshal	176
6.11.3.4	looseUnmarshal	177
6.11.3.5	tightMarshal1	177
6.11.3.6	tightMarshal2	177
6.11.3.7	tightUnmarshal	178
6.12	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	
	Class Reference	178
6.12.1	Detailed Description	179
6.12.2	Constructor & Destructor Documentation	180

6.12.2.1	ActiveMQBlobMessageMarshaller	180
6.12.2.2	~ActiveMQBlobMessageMarshaller	180
6.12.3	Member Function Documentation	180
6.12.3.1	createObject	180
6.12.3.2	getDataStructureType	180
6.12.3.3	looseMarshal	180
6.12.3.4	looseUnmarshal	181
6.12.3.5	tightMarshal1	181
6.12.3.6	tightMarshal2	181
6.12.3.7	tightUnmarshal	182
6.13	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller	
	Class Reference	182
6.13.1	Detailed Description	183
6.13.2	Constructor & Destructor Documentation	184
6.13.2.1	ActiveMQBlobMessageMarshaller	184
6.13.2.2	~ActiveMQBlobMessageMarshaller	184
6.13.3	Member Function Documentation	184
6.13.3.1	createObject	184
6.13.3.2	getDataStructureType	184
6.13.3.3	looseMarshal	184
6.13.3.4	looseUnmarshal	185
6.13.3.5	tightMarshal1	185
6.13.3.6	tightMarshal2	185
6.13.3.7	tightUnmarshal	186
6.14	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	
	Class Reference	186
6.14.1	Detailed Description	187
6.14.2	Constructor & Destructor Documentation	188
6.14.2.1	ActiveMQBlobMessageMarshaller	188
6.14.2.2	~ActiveMQBlobMessageMarshaller	188
6.14.3	Member Function Documentation	188
6.14.3.1	createObject	188
6.14.3.2	getDataStructureType	188
6.14.3.3	looseMarshal	188
6.14.3.4	looseUnmarshal	189
6.14.3.5	tightMarshal1	189
6.14.3.6	tightMarshal2	189

6.14.3.7	tightUnmarshal	190
6.15	activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller	
	Class Reference	190
6.15.1	Detailed Description	191
6.15.2	Constructor & Destructor Documentation	192
6.15.2.1	ActiveMQBlobMessageMarshaller	192
6.15.2.2	~ActiveMQBlobMessageMarshaller	192
6.15.3	Member Function Documentation	192
6.15.3.1	createObject	192
6.15.3.2	getDataStructureType	192
6.15.3.3	looseMarshal	192
6.15.3.4	looseUnmarshal	193
6.15.3.5	tightMarshal1	193
6.15.3.6	tightMarshal2	193
6.15.3.7	tightUnmarshal	194
6.16	activemq::commands::ActiveMQBytesMessage Class Reference	194
6.16.1	Constructor & Destructor Documentation	198
6.16.1.1	ActiveMQBytesMessage	198
6.16.1.2	~ActiveMQBytesMessage	198
6.16.2	Member Function Documentation	198
6.16.2.1	clearBody	198
6.16.2.2	clone	198
6.16.2.3	cloneDataStructure	198
6.16.2.4	copyDataStructure	198
6.16.2.5	equals	199
6.16.2.6	getBodyBytes	199
6.16.2.7	getBodyLength	199
6.16.2.8	getDataStructureType	200
6.16.2.9	onSend	200
6.16.2.10	readBoolean	200
6.16.2.11	readByte	200
6.16.2.12	readBytes	201
6.16.2.13	readBytes	201
6.16.2.14	readChar	202
6.16.2.15	readDouble	202
6.16.2.16	readFloat	203
6.16.2.17	readInt	203

6.16.2.18 readLong	203
6.16.2.19 readShort	204
6.16.2.20 readString	204
6.16.2.21 readUnsignedShort	204
6.16.2.22 readUTF	205
6.16.2.23 reset	205
6.16.2.24 setBodyBytes	205
6.16.2.25 toString	205
6.16.2.26 writeBoolean	206
6.16.2.27 writeByte	206
6.16.2.28 writeBytes	206
6.16.2.29 writeBytes	207
6.16.2.30 writeChar	207
6.16.2.31 writeDouble	207
6.16.2.32 writeFloat	207
6.16.2.33 writeInt	208
6.16.2.34 writeLong	208
6.16.2.35 writeShort	208
6.16.2.36 writeString	209
6.16.2.37 writeUnsignedShort	209
6.16.2.38 writeUTF	209
6.16.3 Field Documentation	209
6.16.3.1 ID_ACTIVEMQBYTESMESSAGE	209
6.17 activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller	
Class Reference	210
6.17.1 Detailed Description	211
6.17.2 Constructor & Destructor Documentation	211
6.17.2.1 ActiveMQBytesMessageMarshaller	211
6.17.2.2 ~ActiveMQBytesMessageMarshaller	211
6.17.3 Member Function Documentation	211
6.17.3.1 createObject	211
6.17.3.2 getDataStructureType	211
6.17.3.3 looseMarshal	211
6.17.3.4 looseUnmarshal	212
6.17.3.5 tightMarshal1	212
6.17.3.6 tightMarshal2	213
6.17.3.7 tightUnmarshal	213

6.18	activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	
	Class Reference	213
6.18.1	Detailed Description	214
6.18.2	Constructor & Destructor Documentation	215
	6.18.2.1 ActiveMQBytesMessageMarshaller	215
	6.18.2.2 ~ActiveMQBytesMessageMarshaller	215
6.18.3	Member Function Documentation	215
	6.18.3.1 createObject	215
	6.18.3.2 getDataStructureType	215
	6.18.3.3 looseMarshal	215
	6.18.3.4 looseUnmarshal	216
	6.18.3.5 tightMarshal1	216
	6.18.3.6 tightMarshal2	216
	6.18.3.7 tightUnmarshal	217
6.19	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller	
	Class Reference	217
6.19.1	Detailed Description	218
6.19.2	Constructor & Destructor Documentation	219
	6.19.2.1 ActiveMQBytesMessageMarshaller	219
	6.19.2.2 ~ActiveMQBytesMessageMarshaller	219
6.19.3	Member Function Documentation	219
	6.19.3.1 createObject	219
	6.19.3.2 getDataStructureType	219
	6.19.3.3 looseMarshal	219
	6.19.3.4 looseUnmarshal	220
	6.19.3.5 tightMarshal1	220
	6.19.3.6 tightMarshal2	220
	6.19.3.7 tightUnmarshal	221
6.20	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	
	Class Reference	221
6.20.1	Detailed Description	222
6.20.2	Constructor & Destructor Documentation	223
	6.20.2.1 ActiveMQBytesMessageMarshaller	223
	6.20.2.2 ~ActiveMQBytesMessageMarshaller	223
6.20.3	Member Function Documentation	223
	6.20.3.1 createObject	223
	6.20.3.2 getDataStructureType	223

6.20.3.3	looseMarshal	223
6.20.3.4	looseUnmarshal	224
6.20.3.5	tightMarshal1	224
6.20.3.6	tightMarshal2	224
6.20.3.7	tightUnmarshal	225
6.21	activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller	
	Class Reference	225
6.21.1	Detailed Description	226
6.21.2	Constructor & Destructor Documentation	227
6.21.2.1	ActiveMQBytesMessageMarshaller	227
6.21.2.2	~ActiveMQBytesMessageMarshaller	227
6.21.3	Member Function Documentation	227
6.21.3.1	createObject	227
6.21.3.2	getDataStructureType	227
6.21.3.3	looseMarshal	227
6.21.3.4	looseUnmarshal	228
6.21.3.5	tightMarshal1	228
6.21.3.6	tightMarshal2	228
6.21.3.7	tightUnmarshal	229
6.22	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller	
	Class Reference	229
6.22.1	Detailed Description	230
6.22.2	Constructor & Destructor Documentation	231
6.22.2.1	ActiveMQBytesMessageMarshaller	231
6.22.2.2	~ActiveMQBytesMessageMarshaller	231
6.22.3	Member Function Documentation	231
6.22.3.1	createObject	231
6.22.3.2	getDataStructureType	231
6.22.3.3	looseMarshal	231
6.22.3.4	looseUnmarshal	232
6.22.3.5	tightMarshal1	232
6.22.3.6	tightMarshal2	232
6.22.3.7	tightUnmarshal	233
6.23	activemq::core::ActiveMQConnection Class Reference	233
6.23.1	Detailed Description	238
6.23.2	Constructor & Destructor Documentation	238
6.23.2.1	ActiveMQConnection	238

6.23.2.2	~ActiveMQConnection	238
6.23.3	Member Function Documentation	238
6.23.3.1	addDispatcher	238
6.23.3.2	addProducer	239
6.23.3.3	addTransportListener	239
6.23.3.4	close	239
6.23.3.5	createSession	239
6.23.3.6	createSession	240
6.23.3.7	destroyDestination	240
6.23.3.8	destroyDestination	240
6.23.3.9	fire	241
6.23.3.10	getBrokerURL	241
6.23.3.11	getClientID	241
6.23.3.12	getCloseTimeout	241
6.23.3.13	getConnectionId	241
6.23.3.14	getConnectionInfo	242
6.23.3.15	getExceptionListener	242
6.23.3.16	getMetaData	242
6.23.3.17	getNextLocalTransactionId	242
6.23.3.18	getNextSessionId	243
6.23.3.19	getNextTempDestinationId	243
6.23.3.20	getPassword	243
6.23.3.21	getPrefetchPolicy	243
6.23.3.22	getProducerWindowSize	243
6.23.3.23	getRedeliveryPolicy	244
6.23.3.24	getSendTimeout	244
6.23.3.25	getTransport	244
6.23.3.26	getUsername	244
6.23.3.27	isAlwaysSyncSend	244
6.23.3.28	isClosed	245
6.23.3.29	isDispatchAsync	245
6.23.3.30	isStarted	245
6.23.3.31	isTransportFailed	245
6.23.3.32	isUseAsyncSend	245
6.23.3.33	isUseCompression	245
6.23.3.34	onCommand	246

6.23.3.35 oneway	246
6.23.3.36 onException	246
6.23.3.37 removeDispatcher	246
6.23.3.38 removeProducer	246
6.23.3.39 removeSession	247
6.23.3.40 removeTransportListener	247
6.23.3.41 sendPullRequest	247
6.23.3.42 setAlwaysSyncSend	247
6.23.3.43 setBrokerURL	247
6.23.3.44 setClientID	248
6.23.3.45 setCloseTimeout	248
6.23.3.46 setDefaultClientId	248
6.23.3.47 setDispatchAsync	248
6.23.3.48 setExceptionListener	248
6.23.3.49 setPassword	248
6.23.3.50 setPrefetchPolicy	249
6.23.3.51 setProducerWindowSize	249
6.23.3.52 setRedeliveryPolicy	249
6.23.3.53 setSendTimeout	249
6.23.3.54 setTransportInterruptionProcessingComplete	249
6.23.3.55 setUseAsyncSend	250
6.23.3.56 setUseCompression	250
6.23.3.57 setUsername	250
6.23.3.58 start	250
6.23.3.59 stop	250
6.23.3.60 syncRequest	251
6.23.3.61 transportInterrupted	251
6.23.3.62 transportResumed	251
6.24 activemq::core::ActiveMQConnectionFactory Class Reference	251
6.24.1 Constructor & Destructor Documentation	254
6.24.1.1 ActiveMQConnectionFactory	254
6.24.1.2 ActiveMQConnectionFactory	254
6.24.1.3 ~ActiveMQConnectionFactory	254
6.24.2 Member Function Documentation	254
6.24.2.1 createConnection	254
6.24.2.2 createConnection	255

6.24.2.3	createConnection	255
6.24.2.4	createConnection	255
6.24.2.5	getBrokerURL	256
6.24.2.6	getClientId	256
6.24.2.7	getCloseTimeout	256
6.24.2.8	getExceptionListener	257
6.24.2.9	getPassword	257
6.24.2.10	getPrefetchPolicy	257
6.24.2.11	getProducerWindowSize	257
6.24.2.12	getRedeliveryPolicy	257
6.24.2.13	getSendTimeout	258
6.24.2.14	getUsername	258
6.24.2.15	isAlwaysSyncSend	258
6.24.2.16	isDispatchAsync	258
6.24.2.17	isUseAsyncSend	258
6.24.2.18	isUseCompression	258
6.24.2.19	setAlwaysSyncSend	259
6.24.2.20	setBrokerURL	259
6.24.2.21	setClientId	259
6.24.2.22	setCloseTimeout	259
6.24.2.23	setDispatchAsync	259
6.24.2.24	setExceptionListener	260
6.24.2.25	setPassword	260
6.24.2.26	setPrefetchPolicy	260
6.24.2.27	setProducerWindowSize	260
6.24.2.28	setRedeliveryPolicy	260
6.24.2.29	setSendTimeout	261
6.24.2.30	setUseAsyncSend	261
6.24.2.31	setUseCompression	261
6.24.2.32	setUsername	261
6.24.3	Field Documentation	261
6.24.3.1	DEFAULT_URI	261
6.25	activemq::core::ActiveMQConnectionMetaData Class Reference	262
6.25.1	Detailed Description	262
6.25.2	Constructor & Destructor Documentation	263
6.25.2.1	ActiveMQConnectionMetaData	263

6.25.2.2	~ActiveMQConnectionMetaData	263
6.25.3	Member Function Documentation	263
6.25.3.1	getCMSMajorVersion	263
6.25.3.2	getCMSMinorVersion	263
6.25.3.3	getCMSProviderName	263
6.25.3.4	getCMSVersion	264
6.25.3.5	getCMSXPropertyNames	264
6.25.3.6	getProviderMajorVersion	264
6.25.3.7	getProviderMinorVersion	265
6.25.3.8	getProviderVersion	265
6.26	activemq::core::ActiveMQConstants Class Reference	265
6.26.1	Detailed Description	266
6.26.2	Member Enumeration Documentation	267
6.26.2.1	AckType	267
6.26.2.2	DestinationActions	267
6.26.2.3	DestinationOption	267
6.26.2.4	TransactionState	267
6.26.2.5	URIParam	268
6.26.3	Member Function Documentation	268
6.26.3.1	toDestinationOption	268
6.26.3.2	toString	268
6.26.3.3	toString	268
6.26.3.4	toURIOption	268
6.27	activemq::core::ActiveMQConsumer Class Reference	268
6.27.1	Constructor & Destructor Documentation	271
6.27.1.1	ActiveMQConsumer	271
6.27.1.2	~ActiveMQConsumer	271
6.27.2	Member Function Documentation	271
6.27.2.1	acknowledge	271
6.27.2.2	acknowledge	271
6.27.2.3	afterMessageIsConsumed	272
6.27.2.4	beforeMessageIsConsumed	272
6.27.2.5	clearMessagesInProgress	272
6.27.2.6	close	272
6.27.2.7	commit	272
6.27.2.8	deliverAcks	273

6.27.2.9	dequeue	273
6.27.2.10	dispatch	273
6.27.2.11	doClose	273
6.27.2.12	getConsumerId	273
6.27.2.13	getConsumerInfo	274
6.27.2.14	getLastDeliveredSequenceId	274
6.27.2.15	getMessageAvailableCount	274
6.27.2.16	getMessageListener	274
6.27.2.17	getMessageSelector	274
6.27.2.18	getRedeliveryPolicy	275
6.27.2.19	inProgressClearRequired	275
6.27.2.20	isClosed	275
6.27.2.21	isSynchronizationRegistered	275
6.27.2.22	iterate	275
6.27.2.23	receive	275
6.27.2.24	receive	276
6.27.2.25	receiveNoWait	276
6.27.2.26	rollback	276
6.27.2.27	setLastDeliveredSequenceId	276
6.27.2.28	setMessageListener	277
6.27.2.29	setRedeliveryPolicy	277
6.27.2.30	setSynchronizationRegistered	277
6.27.2.31	start	277
6.27.2.32	stop	277
6.28	activemq::library::ActiveMQCPP Class Reference	277
6.28.1	Constructor & Destructor Documentation	278
6.28.1.1	ActiveMQCPP	278
6.28.1.2	ActiveMQCPP	278
6.28.1.3	~ActiveMQCPP	278
6.28.2	Member Function Documentation	278
6.28.2.1	initializeLibrary	278
6.28.2.2	initializeLibrary	279
6.28.2.3	operator=	279
6.28.2.4	shutdownLibrary	279
6.29	activemq::commands::ActiveMQDestination Class Reference	279
6.29.1	Constructor & Destructor Documentation	282

6.29.1.1	ActiveMQDestination	282
6.29.1.2	ActiveMQDestination	282
6.29.1.3	~ActiveMQDestination	282
6.29.2	Member Function Documentation	282
6.29.2.1	cloneDataStructure	282
6.29.2.2	copyDataStructure	282
6.29.2.3	createDestination	283
6.29.2.4	createTemporaryName	283
6.29.2.5	equals	283
6.29.2.6	getClientId	283
6.29.2.7	getCMSDestination	284
6.29.2.8	getDataStructureType	284
6.29.2.9	getDestinationType	284
6.29.2.10	getOptions	284
6.29.2.11	getOrderedTarget	284
6.29.2.12	getPhysicalName	285
6.29.2.13	getPhysicalName	285
6.29.2.14	isAdvisory	285
6.29.2.15	isComposite	285
6.29.2.16	isConnectionAdvisory	285
6.29.2.17	isConsumerAdvisory	285
6.29.2.18	isExclusive	286
6.29.2.19	isOrdered	286
6.29.2.20	isProducerAdvisory	286
6.29.2.21	isQueue	286
6.29.2.22	isTemporary	286
6.29.2.23	isTopic	286
6.29.2.24	isWildcard	287
6.29.2.25	setAdvisory	287
6.29.2.26	setExclusive	287
6.29.2.27	setOrdered	287
6.29.2.28	setOrderedTarget	287
6.29.2.29	setPhysicalName	287
6.29.2.30	toString	288
6.29.3	Field Documentation	288
6.29.3.1	advisory	288

6.29.3.2	ADVISORY_PREFIX	288
6.29.3.3	COMPOSITE_SEPARATOR	288
6.29.3.4	CONNECTION_ADVISORY_PREFIX	288
6.29.3.5	CONSUMER_ADVISORY_PREFIX	288
6.29.3.6	DEFAULT_ORDERED_TARGET	288
6.29.3.7	exclusive	289
6.29.3.8	ID_ACTIVEMQDESTINATION	289
6.29.3.9	options	289
6.29.3.10	ordered	289
6.29.3.11	orderedTarget	289
6.29.3.12	physicalName	289
6.29.3.13	PRODUCER_ADVISORY_PREFIX	289
6.29.3.14	QUEUE_QUALIFIED_PREFIX	289
6.29.3.15	TEMP_POSTFIX	289
6.29.3.16	TEMP_PREFIX	289
6.29.3.17	TEMP_QUEUE_QUALIFIED_PREFIX	289
6.29.3.18	TEMP_TOPIC_QUALIFIED_PREFIX	289
6.29.3.19	TOPIC_QUALIFIED_PREFIX	289
6.30	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	
	Class Reference	289
6.30.1	Detailed Description	290
6.30.2	Constructor & Destructor Documentation	291
6.30.2.1	ActiveMQDestinationMarshaller	291
6.30.2.2	~ActiveMQDestinationMarshaller	291
6.30.3	Member Function Documentation	291
6.30.3.1	looseMarshal	291
6.30.3.2	looseUnmarshal	291
6.30.3.3	tightMarshal1	292
6.30.3.4	tightMarshal2	292
6.30.3.5	tightUnmarshal	293
6.31	activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller	
	Class Reference	293
6.31.1	Detailed Description	294
6.31.2	Constructor & Destructor Documentation	295
6.31.2.1	ActiveMQDestinationMarshaller	295
6.31.2.2	~ActiveMQDestinationMarshaller	295
6.31.3	Member Function Documentation	295

6.31.3.1	looseMarshal	295
6.31.3.2	looseUnmarshal	295
6.31.3.3	tightMarshal1	296
6.31.3.4	tightMarshal2	296
6.31.3.5	tightUnmarshal	297
6.32	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	
	Class Reference	297
6.32.1	Detailed Description	298
6.32.2	Constructor & Destructor Documentation	299
6.32.2.1	ActiveMQDestinationMarshaller	299
6.32.2.2	~ActiveMQDestinationMarshaller	299
6.32.3	Member Function Documentation	299
6.32.3.1	looseMarshal	299
6.32.3.2	looseUnmarshal	299
6.32.3.3	tightMarshal1	300
6.32.3.4	tightMarshal2	300
6.32.3.5	tightUnmarshal	301
6.33	activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	
	Class Reference	301
6.33.1	Detailed Description	302
6.33.2	Constructor & Destructor Documentation	303
6.33.2.1	ActiveMQDestinationMarshaller	303
6.33.2.2	~ActiveMQDestinationMarshaller	303
6.33.3	Member Function Documentation	303
6.33.3.1	looseMarshal	303
6.33.3.2	looseUnmarshal	303
6.33.3.3	tightMarshal1	304
6.33.3.4	tightMarshal2	304
6.33.3.5	tightUnmarshal	305
6.34	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	
	Class Reference	305
6.34.1	Detailed Description	306
6.34.2	Constructor & Destructor Documentation	307
6.34.2.1	ActiveMQDestinationMarshaller	307
6.34.2.2	~ActiveMQDestinationMarshaller	307
6.34.3	Member Function Documentation	307
6.34.3.1	looseMarshal	307

6.34.3.2	looseUnmarshal	307
6.34.3.3	tightMarshal1	308
6.34.3.4	tightMarshal2	308
6.34.3.5	tightUnmarshal	309
6.35	activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller Class Reference	309
6.35.1	Detailed Description	310
6.35.2	Constructor & Destructor Documentation	311
6.35.2.1	ActiveMQDestinationMarshaller	311
6.35.2.2	~ActiveMQDestinationMarshaller	311
6.35.3	Member Function Documentation	311
6.35.3.1	looseMarshal	311
6.35.3.2	looseUnmarshal	311
6.35.3.3	tightMarshal1	312
6.35.3.4	tightMarshal2	312
6.35.3.5	tightUnmarshal	313
6.36	activemq::exceptions::ActiveMQException Class Reference	313
6.36.1	Constructor & Destructor Documentation	314
6.36.1.1	ActiveMQException	314
6.36.1.2	ActiveMQException	314
6.36.1.3	ActiveMQException	314
6.36.1.4	ActiveMQException	315
6.36.1.5	~ActiveMQException	315
6.36.2	Member Function Documentation	315
6.36.2.1	clone	315
6.36.2.2	convertToCMSException	315
6.37	activemq::commands::ActiveMQMapMessage Class Reference	316
6.37.1	Constructor & Destructor Documentation	319
6.37.1.1	ActiveMQMapMessage	319
6.37.1.2	~ActiveMQMapMessage	319
6.37.2	Member Function Documentation	319
6.37.2.1	beforeMarshal	319
6.37.2.2	checkMapIsUnmarshalled	319
6.37.2.3	clearBody	319
6.37.2.4	clone	319
6.37.2.5	cloneDataStructure	320
6.37.2.6	copyDataStructure	320

6.37.2.7	<code>equals</code>	320
6.37.2.8	<code>getBoolean</code>	320
6.37.2.9	<code>getByte</code>	321
6.37.2.10	<code>getBytes</code>	321
6.37.2.11	<code>getChar</code>	321
6.37.2.12	<code>getDataStructureType</code>	322
6.37.2.13	<code>getDouble</code>	322
6.37.2.14	<code>getFloat</code>	322
6.37.2.15	<code>getInt</code>	322
6.37.2.16	<code>getLong</code>	323
6.37.2.17	<code>getMap</code>	323
6.37.2.18	<code>getMap</code>	323
6.37.2.19	<code>getMapNames</code>	323
6.37.2.20	<code>getShort</code>	324
6.37.2.21	<code>getString</code>	324
6.37.2.22	<code>isMarshalAware</code>	324
6.37.2.23	<code>itemExists</code>	324
6.37.2.24	<code>setBoolean</code>	325
6.37.2.25	<code>setByte</code>	325
6.37.2.26	<code>setBytes</code>	325
6.37.2.27	<code>setChar</code>	326
6.37.2.28	<code>setDouble</code>	326
6.37.2.29	<code>setFloat</code>	326
6.37.2.30	<code>setInt</code>	327
6.37.2.31	<code>setLong</code>	327
6.37.2.32	<code>setShort</code>	327
6.37.2.33	<code>setString</code>	328
6.37.2.34	<code>toString</code>	328
6.37.3	Field Documentation	328
6.37.3.1	<code>ID_ACTIVEMQMAPMESSAGE</code>	328
6.38	<code>activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller</code>	
	Class Reference	328
6.38.1	Detailed Description	329
6.38.2	Constructor & Destructor Documentation	330
6.38.2.1	<code>ActiveMQMapMessageMarshaller</code>	330
6.38.2.2	<code>~ActiveMQMapMessageMarshaller</code>	330
6.38.3	Member Function Documentation	330

6.38.3.1	createObject	330
6.38.3.2	getDataStructureType	330
6.38.3.3	looseMarshal	330
6.38.3.4	looseUnmarshal	331
6.38.3.5	tightMarshal1	331
6.38.3.6	tightMarshal2	331
6.38.3.7	tightUnmarshal	332
6.39	activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	
	Class Reference	332
6.39.1	Detailed Description	333
6.39.2	Constructor & Destructor Documentation	334
6.39.2.1	ActiveMQMapMessageMarshaller	334
6.39.2.2	~ActiveMQMapMessageMarshaller	334
6.39.3	Member Function Documentation	334
6.39.3.1	createObject	334
6.39.3.2	getDataStructureType	334
6.39.3.3	looseMarshal	334
6.39.3.4	looseUnmarshal	335
6.39.3.5	tightMarshal1	335
6.39.3.6	tightMarshal2	335
6.39.3.7	tightUnmarshal	336
6.40	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	
	Class Reference	336
6.40.1	Detailed Description	337
6.40.2	Constructor & Destructor Documentation	338
6.40.2.1	ActiveMQMapMessageMarshaller	338
6.40.2.2	~ActiveMQMapMessageMarshaller	338
6.40.3	Member Function Documentation	338
6.40.3.1	createObject	338
6.40.3.2	getDataStructureType	338
6.40.3.3	looseMarshal	338
6.40.3.4	looseUnmarshal	339
6.40.3.5	tightMarshal1	339
6.40.3.6	tightMarshal2	339
6.40.3.7	tightUnmarshal	340
6.41	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	
	Class Reference	340

6.41.1	Detailed Description	341
6.41.2	Constructor & Destructor Documentation	342
6.41.2.1	ActiveMQMapMessageMarshaller	342
6.41.2.2	~ActiveMQMapMessageMarshaller	342
6.41.3	Member Function Documentation	342
6.41.3.1	createObject	342
6.41.3.2	getDataStructureType	342
6.41.3.3	looseMarshal	342
6.41.3.4	looseUnmarshal	343
6.41.3.5	tightMarshal1	343
6.41.3.6	tightMarshal2	343
6.41.3.7	tightUnmarshal	344
6.42	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller	
	Class Reference	344
6.42.1	Detailed Description	345
6.42.2	Constructor & Destructor Documentation	346
6.42.2.1	ActiveMQMapMessageMarshaller	346
6.42.2.2	~ActiveMQMapMessageMarshaller	346
6.42.3	Member Function Documentation	346
6.42.3.1	createObject	346
6.42.3.2	getDataStructureType	346
6.42.3.3	looseMarshal	346
6.42.3.4	looseUnmarshal	347
6.42.3.5	tightMarshal1	347
6.42.3.6	tightMarshal2	347
6.42.3.7	tightUnmarshal	348
6.43	activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	
	Class Reference	348
6.43.1	Detailed Description	349
6.43.2	Constructor & Destructor Documentation	350
6.43.2.1	ActiveMQMapMessageMarshaller	350
6.43.2.2	~ActiveMQMapMessageMarshaller	350
6.43.3	Member Function Documentation	350
6.43.3.1	createObject	350
6.43.3.2	getDataStructureType	350
6.43.3.3	looseMarshal	350
6.43.3.4	looseUnmarshal	351

6.43.3.5	tightMarshal1	351
6.43.3.6	tightMarshal2	351
6.43.3.7	tightUnmarshal	352
6.44	activemq::commands::ActiveMQMessage Class Reference	352
6.44.1	Constructor & Destructor Documentation	353
6.44.1.1	ActiveMQMessage	353
6.44.1.2	~ActiveMQMessage	353
6.44.2	Member Function Documentation	353
6.44.2.1	clone	353
6.44.2.2	cloneDataStructure	354
6.44.2.3	copyDataStructure	354
6.44.2.4	equals	354
6.44.2.5	getDataStructureType	354
6.44.2.6	toString	355
6.44.3	Field Documentation	355
6.44.3.1	ID_ACTIVEMQMESSAGE	355
6.45	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller Class Reference	355
6.45.1	Detailed Description	356
6.45.2	Constructor & Destructor Documentation	356
6.45.2.1	ActiveMQMessageMarshaller	356
6.45.2.2	~ActiveMQMessageMarshaller	356
6.45.3	Member Function Documentation	356
6.45.3.1	createObject	356
6.45.3.2	getDataStructureType	356
6.45.3.3	looseMarshal	357
6.45.3.4	looseUnmarshal	357
6.45.3.5	tightMarshal1	357
6.45.3.6	tightMarshal2	358
6.45.3.7	tightUnmarshal	358
6.46	activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller Class Reference	359
6.46.1	Detailed Description	360
6.46.2	Constructor & Destructor Documentation	360
6.46.2.1	ActiveMQMessageMarshaller	360
6.46.2.2	~ActiveMQMessageMarshaller	360
6.46.3	Member Function Documentation	360

6.46.3.1	createObject	360
6.46.3.2	getDataStructureType	360
6.46.3.3	looseMarshal	361
6.46.3.4	looseUnmarshal	361
6.46.3.5	tightMarshal1	361
6.46.3.6	tightMarshal2	362
6.46.3.7	tightUnmarshal	362
6.47	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller Class Reference	363
6.47.1	Detailed Description	364
6.47.2	Constructor & Destructor Documentation	364
6.47.2.1	ActiveMQMessageMarshaller	364
6.47.2.2	~ActiveMQMessageMarshaller	364
6.47.3	Member Function Documentation	364
6.47.3.1	createObject	364
6.47.3.2	getDataStructureType	364
6.47.3.3	looseMarshal	365
6.47.3.4	looseUnmarshal	365
6.47.3.5	tightMarshal1	365
6.47.3.6	tightMarshal2	366
6.47.3.7	tightUnmarshal	366
6.48	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller Class Reference	367
6.48.1	Detailed Description	368
6.48.2	Constructor & Destructor Documentation	368
6.48.2.1	ActiveMQMessageMarshaller	368
6.48.2.2	~ActiveMQMessageMarshaller	368
6.48.3	Member Function Documentation	368
6.48.3.1	createObject	368
6.48.3.2	getDataStructureType	368
6.48.3.3	looseMarshal	369
6.48.3.4	looseUnmarshal	369
6.48.3.5	tightMarshal1	369
6.48.3.6	tightMarshal2	370
6.48.3.7	tightUnmarshal	370
6.49	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller Class Reference	371

6.49.1	Detailed Description	372
6.49.2	Constructor & Destructor Documentation	372
6.49.2.1	ActiveMQMessageMarshaller	372
6.49.2.2	~ActiveMQMessageMarshaller	372
6.49.3	Member Function Documentation	372
6.49.3.1	createObject	372
6.49.3.2	getDataStructureType	372
6.49.3.3	looseMarshal	373
6.49.3.4	looseUnmarshal	373
6.49.3.5	tightMarshal1	373
6.49.3.6	tightMarshal2	374
6.49.3.7	tightUnmarshal	374
6.50	activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller Class Reference	375
6.50.1	Detailed Description	376
6.50.2	Constructor & Destructor Documentation	376
6.50.2.1	ActiveMQMessageMarshaller	376
6.50.2.2	~ActiveMQMessageMarshaller	376
6.50.3	Member Function Documentation	376
6.50.3.1	createObject	376
6.50.3.2	getDataStructureType	376
6.50.3.3	looseMarshal	377
6.50.3.4	looseUnmarshal	377
6.50.3.5	tightMarshal1	377
6.50.3.6	tightMarshal2	378
6.50.3.7	tightUnmarshal	378
6.51	activemq::commands::ActiveMQMessageTemplate< T > Class Template Reference	379
6.51.1	Constructor & Destructor Documentation	383
6.51.1.1	ActiveMQMessageTemplate	383
6.51.1.2	~ActiveMQMessageTemplate	383
6.51.2	Member Function Documentation	383
6.51.2.1	acknowledge	383
6.51.2.2	clearBody	383
6.51.2.3	clearProperties	383
6.51.2.4	equals	383
6.51.2.5	failIfReadOnlyBody	384
6.51.2.6	failIfReadOnlyProperties	384

6.51.2.7 failIfWriteOnlyBody	384
6.51.2.8 getBooleanProperty	384
6.51.2.9 getByteProperty	384
6.51.2.10 getCMSCorrelationID	385
6.51.2.11 getCMSDeliveryMode	385
6.51.2.12 getCMSDestination	385
6.51.2.13 getCMSExpiration	386
6.51.2.14 getCMSMessageID	386
6.51.2.15 getCMSPriority	386
6.51.2.16 getCMSRedelivered	386
6.51.2.17 getCMSReplyTo	387
6.51.2.18 getCMSTimestamp	387
6.51.2.19 getCMSType	387
6.51.2.20 getDoubleProperty	388
6.51.2.21 getFloatProperty	388
6.51.2.22 getIntProperty	388
6.51.2.23 getLongProperty	389
6.51.2.24 getPropertyNames	389
6.51.2.25 getShortProperty	389
6.51.2.26 getStringProperty	390
6.51.2.27 onSend	390
6.51.2.28 propertyExists	390
6.51.2.29 setBooleanProperty	390
6.51.2.30 setByteProperty	391
6.51.2.31 setCMSCorrelationID	391
6.51.2.32 setCMSDeliveryMode	391
6.51.2.33 setCMSDestination	392
6.51.2.34 setCMSExpiration	392
6.51.2.35 setCMSMessageID	392
6.51.2.36 setCMSPriority	393
6.51.2.37 setCMSRedelivered	393
6.51.2.38 setCMSReplyTo	393
6.51.2.39 setCMSTimestamp	393
6.51.2.40 setCMSType	394
6.51.2.41 setDoubleProperty	394
6.51.2.42 setFloatProperty	394

6.51.2.43	setIntProperty	395
6.51.2.44	setLongProperty	395
6.51.2.45	setShortProperty	395
6.51.2.46	setStringProperty	396
6.52	activemq::commands::ActiveMQObjectMessage Class Reference	396
6.52.1	Constructor & Destructor Documentation	397
6.52.1.1	ActiveMQObjectMessage	397
6.52.1.2	~ActiveMQObjectMessage	397
6.52.2	Member Function Documentation	397
6.52.2.1	clone	397
6.52.2.2	cloneDataStructure	397
6.52.2.3	copyDataStructure	398
6.52.2.4	equals	398
6.52.2.5	getDataStructureType	398
6.52.2.6	toString	398
6.52.3	Field Documentation	399
6.52.3.1	ID_ACTIVEMQOBJECTMESSAGE	399
6.53	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller Class Reference	399
6.53.1	Detailed Description	400
6.53.2	Constructor & Destructor Documentation	400
6.53.2.1	ActiveMQObjectMessageMarshaller	400
6.53.2.2	~ActiveMQObjectMessageMarshaller	400
6.53.3	Member Function Documentation	400
6.53.3.1	createObject	400
6.53.3.2	getDataStructureType	400
6.53.3.3	looseMarshal	401
6.53.3.4	looseUnmarshal	401
6.53.3.5	tightMarshal1	401
6.53.3.6	tightMarshal2	402
6.53.3.7	tightUnmarshal	402
6.54	activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller Class Reference	403
6.54.1	Detailed Description	404
6.54.2	Constructor & Destructor Documentation	404
6.54.2.1	ActiveMQObjectMessageMarshaller	404
6.54.2.2	~ActiveMQObjectMessageMarshaller	404

6.54.3	Member Function Documentation	404
6.54.3.1	createObject	404
6.54.3.2	getDataStructureType	404
6.54.3.3	looseMarshal	405
6.54.3.4	looseUnmarshal	405
6.54.3.5	tightMarshal1	405
6.54.3.6	tightMarshal2	406
6.54.3.7	tightUnmarshal	406
6.55	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller Class Reference	407
6.55.1	Detailed Description	408
6.55.2	Constructor & Destructor Documentation	408
6.55.2.1	ActiveMQObjectMessageMarshaller	408
6.55.2.2	~ActiveMQObjectMessageMarshaller	408
6.55.3	Member Function Documentation	408
6.55.3.1	createObject	408
6.55.3.2	getDataStructureType	408
6.55.3.3	looseMarshal	409
6.55.3.4	looseUnmarshal	409
6.55.3.5	tightMarshal1	409
6.55.3.6	tightMarshal2	410
6.55.3.7	tightUnmarshal	410
6.56	activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller Class Reference	411
6.56.1	Detailed Description	412
6.56.2	Constructor & Destructor Documentation	412
6.56.2.1	ActiveMQObjectMessageMarshaller	412
6.56.2.2	~ActiveMQObjectMessageMarshaller	412
6.56.3	Member Function Documentation	412
6.56.3.1	createObject	412
6.56.3.2	getDataStructureType	412
6.56.3.3	looseMarshal	413
6.56.3.4	looseUnmarshal	413
6.56.3.5	tightMarshal1	413
6.56.3.6	tightMarshal2	414
6.56.3.7	tightUnmarshal	414

6.57	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	
	Class Reference	415
6.57.1	Detailed Description	416
6.57.2	Constructor & Destructor Documentation	416
	6.57.2.1 ActiveMQObjectMessageMarshaller	416
	6.57.2.2 ~ActiveMQObjectMessageMarshaller	416
6.57.3	Member Function Documentation	416
	6.57.3.1 createObject	416
	6.57.3.2 getDataStructureType	416
	6.57.3.3 looseMarshal	417
	6.57.3.4 looseUnmarshal	417
	6.57.3.5 tightMarshal1	417
	6.57.3.6 tightMarshal2	418
	6.57.3.7 tightUnmarshal	418
6.58	activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	
	Class Reference	419
6.58.1	Detailed Description	420
6.58.2	Constructor & Destructor Documentation	420
	6.58.2.1 ActiveMQObjectMessageMarshaller	420
	6.58.2.2 ~ActiveMQObjectMessageMarshaller	420
6.58.3	Member Function Documentation	420
	6.58.3.1 createObject	420
	6.58.3.2 getDataStructureType	420
	6.58.3.3 looseMarshal	421
	6.58.3.4 looseUnmarshal	421
	6.58.3.5 tightMarshal1	421
	6.58.3.6 tightMarshal2	422
	6.58.3.7 tightUnmarshal	422
6.59	activemq::core::ActiveMQProducer Class Reference	423
6.59.1	Constructor & Destructor Documentation	425
	6.59.1.1 ActiveMQProducer	425
	6.59.1.2 ~ActiveMQProducer	425
6.59.2	Member Function Documentation	425
	6.59.2.1 close	425
	6.59.2.2 getDeliveryMode	425
	6.59.2.3 getDisableMessageID	425
	6.59.2.4 getDisableMessageTimeStamp	426

6.59.2.5	getPriority	426
6.59.2.6	getProducerId	426
6.59.2.7	getProducerInfo	426
6.59.2.8	getSendTimeout	427
6.59.2.9	getTimeToLive	427
6.59.2.10	isClosed	427
6.59.2.11	onProducerAck	427
6.59.2.12	send	427
6.59.2.13	send	428
6.59.2.14	send	428
6.59.2.15	send	429
6.59.2.16	setDeliveryMode	429
6.59.2.17	setDisableMessageID	430
6.59.2.18	setDisableMessageTimeStamp	430
6.59.2.19	setPriority	430
6.59.2.20	setSendTimeout	430
6.59.2.21	setTimeToLive	430
6.60	activemq::util::ActiveMQProperties Class Reference	431
6.60.1	Detailed Description	432
6.60.2	Constructor & Destructor Documentation	432
6.60.2.1	ActiveMQProperties	432
6.60.2.2	~ActiveMQProperties	432
6.60.3	Member Function Documentation	432
6.60.3.1	clear	432
6.60.3.2	clone	432
6.60.3.3	copy	432
6.60.3.4	getProperties	433
6.60.3.5	getProperties	433
6.60.3.6	getProperty	433
6.60.3.7	getProperty	433
6.60.3.8	hasProperty	433
6.60.3.9	isEmpty	434
6.60.3.10	remove	434
6.60.3.11	setProperties	434
6.60.3.12	setProperty	434
6.60.3.13	toArray	434

6.60.3.14	toString	435
6.61	activemq::commands::ActiveMQQueue Class Reference	435
6.61.1	Constructor & Destructor Documentation	436
6.61.1.1	ActiveMQQueue	436
6.61.1.2	ActiveMQQueue	436
6.61.1.3	~ActiveMQQueue	436
6.61.2	Member Function Documentation	436
6.61.2.1	clone	436
6.61.2.2	cloneDataStructure	436
6.61.2.3	copy	436
6.61.2.4	copyDataStructure	437
6.61.2.5	equals	437
6.61.2.6	getCMSDestination	437
6.61.2.7	getCMSProperties	437
6.61.2.8	getDataStructureType	438
6.61.2.9	getDestinationType	438
6.61.2.10	getQueueName	438
6.61.2.11	toString	438
6.61.3	Field Documentation	438
6.61.3.1	ID_ACTIVEMQQUEUE	438
6.62	activemq::core::ActiveMQQueueBrowser Class Reference	438
6.62.1	Constructor & Destructor Documentation	440
6.62.1.1	ActiveMQQueueBrowser	440
6.62.1.2	~ActiveMQQueueBrowser	440
6.62.2	Member Function Documentation	440
6.62.2.1	close	440
6.62.2.2	getEnumeration	440
6.62.2.3	getMessageSelector	440
6.62.2.4	getQueue	440
6.62.2.5	hasMoreMessages	440
6.62.2.6	nextMessage	440
6.62.3	Friends And Related Function Documentation	441
6.62.3.1	Browser	441
6.63	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller Class Reference	441
6.63.1	Detailed Description	442
6.63.2	Constructor & Destructor Documentation	442

6.63.2.1	ActiveMQQueueMarshaller	442
6.63.2.2	~ActiveMQQueueMarshaller	442
6.63.3	Member Function Documentation	442
6.63.3.1	createObject	442
6.63.3.2	getDataStructureType	442
6.63.3.3	looseMarshal	443
6.63.3.4	looseUnmarshal	443
6.63.3.5	tightMarshal1	443
6.63.3.6	tightMarshal2	444
6.63.3.7	tightUnmarshal	444
6.64	activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller Class	
	Reference	445
6.64.1	Detailed Description	446
6.64.2	Constructor & Destructor Documentation	446
6.64.2.1	ActiveMQQueueMarshaller	446
6.64.2.2	~ActiveMQQueueMarshaller	446
6.64.3	Member Function Documentation	446
6.64.3.1	createObject	446
6.64.3.2	getDataStructureType	446
6.64.3.3	looseMarshal	447
6.64.3.4	looseUnmarshal	447
6.64.3.5	tightMarshal1	447
6.64.3.6	tightMarshal2	448
6.64.3.7	tightUnmarshal	448
6.65	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller Class	
	Reference	449
6.65.1	Detailed Description	450
6.65.2	Constructor & Destructor Documentation	450
6.65.2.1	ActiveMQQueueMarshaller	450
6.65.2.2	~ActiveMQQueueMarshaller	450
6.65.3	Member Function Documentation	450
6.65.3.1	createObject	450
6.65.3.2	getDataStructureType	450
6.65.3.3	looseMarshal	451
6.65.3.4	looseUnmarshal	451
6.65.3.5	tightMarshal1	451
6.65.3.6	tightMarshal2	452

6.65.3.7	tightUnmarshal	452
6.66	activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	Class
	Reference	453
6.66.1	Detailed Description	454
6.66.2	Constructor & Destructor Documentation	454
6.66.2.1	ActiveMQQueueMarshaller	454
6.66.2.2	~ActiveMQQueueMarshaller	454
6.66.3	Member Function Documentation	454
6.66.3.1	createObject	454
6.66.3.2	getDataStructureType	454
6.66.3.3	looseMarshal	455
6.66.3.4	looseUnmarshal	455
6.66.3.5	tightMarshal1	455
6.66.3.6	tightMarshal2	456
6.66.3.7	tightUnmarshal	456
6.67	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	Class
	Reference	457
6.67.1	Detailed Description	458
6.67.2	Constructor & Destructor Documentation	458
6.67.2.1	ActiveMQQueueMarshaller	458
6.67.2.2	~ActiveMQQueueMarshaller	458
6.67.3	Member Function Documentation	458
6.67.3.1	createObject	458
6.67.3.2	getDataStructureType	458
6.67.3.3	looseMarshal	459
6.67.3.4	looseUnmarshal	459
6.67.3.5	tightMarshal1	459
6.67.3.6	tightMarshal2	460
6.67.3.7	tightUnmarshal	460
6.68	activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller	Class
	Reference	461
6.68.1	Detailed Description	462
6.68.2	Constructor & Destructor Documentation	462
6.68.2.1	ActiveMQQueueMarshaller	462
6.68.2.2	~ActiveMQQueueMarshaller	462
6.68.3	Member Function Documentation	462
6.68.3.1	createObject	462

6.68.3.2	getDataStructureType	462
6.68.3.3	looseMarshal	463
6.68.3.4	looseUnmarshal	463
6.68.3.5	tightMarshal1	463
6.68.3.6	tightMarshal2	464
6.68.3.7	tightUnmarshal	464
6.69	activemq::core::ActiveMQSession Class Reference	465
6.69.1	Constructor & Destructor Documentation	469
6.69.1.1	ActiveMQSession	469
6.69.1.2	~ActiveMQSession	469
6.69.2	Member Function Documentation	469
6.69.2.1	acknowledge	469
6.69.2.2	addConsumer	469
6.69.2.3	addProducer	470
6.69.2.4	clearMessagesInProgress	470
6.69.2.5	close	470
6.69.2.6	commit	470
6.69.2.7	createBrowser	470
6.69.2.8	createBrowser	471
6.69.2.9	createBytesMessage	471
6.69.2.10	createBytesMessage	471
6.69.2.11	createConsumer	472
6.69.2.12	createConsumer	472
6.69.2.13	createConsumer	472
6.69.2.14	createDurableConsumer	473
6.69.2.15	createMapMessage	473
6.69.2.16	createMessage	473
6.69.2.17	createProducer	474
6.69.2.18	createQueue	474
6.69.2.19	createStreamMessage	474
6.69.2.20	createTemporaryQueue	474
6.69.2.21	createTemporaryTopic	475
6.69.2.22	createTextMessage	475
6.69.2.23	createTextMessage	475
6.69.2.24	createTopic	475
6.69.2.25	deliverAcks	476

6.69.2.26	dispatch	476
6.69.2.27	doStartTransaction	476
6.69.2.28	fire	476
6.69.2.29	getAcknowledgeMode	476
6.69.2.30	getConnection	476
6.69.2.31	getExceptionListener	477
6.69.2.32	getLastDeliveredSequenceId	477
6.69.2.33	getNextConsumerId	477
6.69.2.34	getNextProducerId	477
6.69.2.35	getSessionId	477
6.69.2.36	getSessionInfo	478
6.69.2.37	getTransactionContext	478
6.69.2.38	isAutoAcknowledge	478
6.69.2.39	isClientAcknowledge	478
6.69.2.40	isDupsOkAcknowledge	478
6.69.2.41	isIndividualAcknowledge	478
6.69.2.42	isStarted	478
6.69.2.43	isTransacted	478
6.69.2.44	oneway	478
6.69.2.45	recover	479
6.69.2.46	redispatch	479
6.69.2.47	removeConsumer	479
6.69.2.48	removeProducer	480
6.69.2.49	rollback	480
6.69.2.50	send	480
6.69.2.51	setLastDeliveredSequenceId	481
6.69.2.52	start	481
6.69.2.53	stop	481
6.69.2.54	syncRequest	481
6.69.2.55	unsubscribe	481
6.69.2.56	wakeup	482
6.69.3	Friends And Related Function Documentation	482
6.69.3.1	ActiveMQSessionExecutor	482
6.70	activemq::core::ActiveMQSessionExecutor Class Reference	482
6.70.1	Detailed Description	483
6.70.2	Constructor & Destructor Documentation	483

6.70.2.1	ActiveMQSessionExecutor	483
6.70.2.2	~ActiveMQSessionExecutor	483
6.70.3	Member Function Documentation	483
6.70.3.1	clear	483
6.70.3.2	clearMessagesInProgress	483
6.70.3.3	close	484
6.70.3.4	execute	484
6.70.3.5	executeFirst	484
6.70.3.6	getUnconsumedMessages	484
6.70.3.7	hasUnconsumedMessages	484
6.70.3.8	isEmpty	484
6.70.3.9	isRunning	485
6.70.3.10	iterate	485
6.70.3.11	start	485
6.70.3.12	stop	485
6.70.3.13	wakeup	485
6.71	activemq::commands::ActiveMQStreamMessage Class Reference	485
6.71.1	Constructor & Destructor Documentation	489
6.71.1.1	ActiveMQStreamMessage	489
6.71.1.2	~ActiveMQStreamMessage	489
6.71.2	Member Function Documentation	489
6.71.2.1	clearBody	489
6.71.2.2	clone	489
6.71.2.3	cloneDataStructure	489
6.71.2.4	copyDataStructure	489
6.71.2.5	equals	490
6.71.2.6	getDataStructureType	490
6.71.2.7	onSend	490
6.71.2.8	readBoolean	490
6.71.2.9	readByte	491
6.71.2.10	readBytes	491
6.71.2.11	readBytes	492
6.71.2.12	readChar	492
6.71.2.13	readDouble	493
6.71.2.14	readFloat	493
6.71.2.15	readInt	493

6.71.2.16	readLong	494
6.71.2.17	readShort	494
6.71.2.18	readString	494
6.71.2.19	readUnsignedShort	495
6.71.2.20	reset	495
6.71.2.21	toString	495
6.71.2.22	writeBoolean	496
6.71.2.23	writeByte	496
6.71.2.24	writeBytes	496
6.71.2.25	writeBytes	497
6.71.2.26	writeChar	497
6.71.2.27	writeDouble	497
6.71.2.28	writeFloat	497
6.71.2.29	writeInt	498
6.71.2.30	writeLong	498
6.71.2.31	writeShort	498
6.71.2.32	writeString	499
6.71.2.33	writeUnsignedShort	499
6.71.3	Field Documentation	499
6.71.3.1	ID_ACTIVEMQSTREAMMESSAGE	499
6.72	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	
	Class Reference	499
6.72.1	Detailed Description	500
6.72.2	Constructor & Destructor Documentation	501
6.72.2.1	ActiveMQStreamMessageMarshaller	501
6.72.2.2	~ActiveMQStreamMessageMarshaller	501
6.72.3	Member Function Documentation	501
6.72.3.1	createObject	501
6.72.3.2	getDataStructureType	501
6.72.3.3	looseMarshal	501
6.72.3.4	looseUnmarshal	502
6.72.3.5	tightMarshal1	502
6.72.3.6	tightMarshal2	502
6.72.3.7	tightUnmarshal	503
6.73	activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	
	Class Reference	503
6.73.1	Detailed Description	504

6.73.2	Constructor & Destructor Documentation	505
6.73.2.1	ActiveMQStreamMessageMarshaller	505
6.73.2.2	~ActiveMQStreamMessageMarshaller	505
6.73.3	Member Function Documentation	505
6.73.3.1	createObject	505
6.73.3.2	getDataStructureType	505
6.73.3.3	looseMarshal	505
6.73.3.4	looseUnmarshal	506
6.73.3.5	tightMarshal1	506
6.73.3.6	tightMarshal2	506
6.73.3.7	tightUnmarshal	507
6.74	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	
	Class Reference	507
6.74.1	Detailed Description	508
6.74.2	Constructor & Destructor Documentation	509
6.74.2.1	ActiveMQStreamMessageMarshaller	509
6.74.2.2	~ActiveMQStreamMessageMarshaller	509
6.74.3	Member Function Documentation	509
6.74.3.1	createObject	509
6.74.3.2	getDataStructureType	509
6.74.3.3	looseMarshal	509
6.74.3.4	looseUnmarshal	510
6.74.3.5	tightMarshal1	510
6.74.3.6	tightMarshal2	510
6.74.3.7	tightUnmarshal	511
6.75	activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	
	Class Reference	511
6.75.1	Detailed Description	512
6.75.2	Constructor & Destructor Documentation	513
6.75.2.1	ActiveMQStreamMessageMarshaller	513
6.75.2.2	~ActiveMQStreamMessageMarshaller	513
6.75.3	Member Function Documentation	513
6.75.3.1	createObject	513
6.75.3.2	getDataStructureType	513
6.75.3.3	looseMarshal	513
6.75.3.4	looseUnmarshal	514
6.75.3.5	tightMarshal1	514

6.75.3.6	tightMarshal2	514
6.75.3.7	tightUnmarshal	515
6.76	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller	
	Class Reference	515
6.76.1	Detailed Description	516
6.76.2	Constructor & Destructor Documentation	517
6.76.2.1	ActiveMQStreamMessageMarshaller	517
6.76.2.2	~ActiveMQStreamMessageMarshaller	517
6.76.3	Member Function Documentation	517
6.76.3.1	createObject	517
6.76.3.2	getDataStructureType	517
6.76.3.3	looseMarshal	517
6.76.3.4	looseUnmarshal	518
6.76.3.5	tightMarshal1	518
6.76.3.6	tightMarshal2	518
6.76.3.7	tightUnmarshal	519
6.77	activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller	
	Class Reference	519
6.77.1	Detailed Description	520
6.77.2	Constructor & Destructor Documentation	521
6.77.2.1	ActiveMQStreamMessageMarshaller	521
6.77.2.2	~ActiveMQStreamMessageMarshaller	521
6.77.3	Member Function Documentation	521
6.77.3.1	createObject	521
6.77.3.2	getDataStructureType	521
6.77.3.3	looseMarshal	521
6.77.3.4	looseUnmarshal	522
6.77.3.5	tightMarshal1	522
6.77.3.6	tightMarshal2	522
6.77.3.7	tightUnmarshal	523
6.78	activemq::commands::ActiveMQTempDestination Class Reference	523
6.78.1	Constructor & Destructor Documentation	525
6.78.1.1	ActiveMQTempDestination	525
6.78.1.2	ActiveMQTempDestination	525
6.78.1.3	~ActiveMQTempDestination	525
6.78.2	Member Function Documentation	525
6.78.2.1	cloneDataStructure	525

6.78.2.2	close	525
6.78.2.3	copyDataStructure	525
6.78.2.4	equals	525
6.78.2.5	getDataStructureType	526
6.78.2.6	setConnection	526
6.78.2.7	toString	526
6.78.3	Field Documentation	526
6.78.3.1	connection	526
6.78.3.2	ID_ACTIVEMQTEMPDESTINATION	526
6.79	activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	
	Class Reference	527
6.79.1	Detailed Description	527
6.79.2	Constructor & Destructor Documentation	528
6.79.2.1	ActiveMQTempDestinationMarshaller	528
6.79.2.2	~ActiveMQTempDestinationMarshaller	528
6.79.3	Member Function Documentation	528
6.79.3.1	looseMarshal	528
6.79.3.2	looseUnmarshal	528
6.79.3.3	tightMarshal1	529
6.79.3.4	tightMarshal2	529
6.79.3.5	tightUnmarshal	530
6.80	activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller	
	Class Reference	530
6.80.1	Detailed Description	531
6.80.2	Constructor & Destructor Documentation	531
6.80.2.1	ActiveMQTempDestinationMarshaller	531
6.80.2.2	~ActiveMQTempDestinationMarshaller	531
6.80.3	Member Function Documentation	531
6.80.3.1	looseMarshal	531
6.80.3.2	looseUnmarshal	532
6.80.3.3	tightMarshal1	532
6.80.3.4	tightMarshal2	533
6.80.3.5	tightUnmarshal	533
6.81	activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	
	Class Reference	534
6.81.1	Detailed Description	535
6.81.2	Constructor & Destructor Documentation	535

6.81.2.1	ActiveMQTempDestinationMarshaller	535
6.81.2.2	~ActiveMQTempDestinationMarshaller	535
6.81.3	Member Function Documentation	535
6.81.3.1	looseMarshal	535
6.81.3.2	looseUnmarshal	536
6.81.3.3	tightMarshal1	536
6.81.3.4	tightMarshal2	537
6.81.3.5	tightUnmarshal	537
6.82	activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller Class Reference	538
6.82.1	Detailed Description	538
6.82.2	Constructor & Destructor Documentation	539
6.82.2.1	ActiveMQTempDestinationMarshaller	539
6.82.2.2	~ActiveMQTempDestinationMarshaller	539
6.82.3	Member Function Documentation	539
6.82.3.1	looseMarshal	539
6.82.3.2	looseUnmarshal	539
6.82.3.3	tightMarshal1	540
6.82.3.4	tightMarshal2	540
6.82.3.5	tightUnmarshal	541
6.83	activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller Class Reference	541
6.83.1	Detailed Description	542
6.83.2	Constructor & Destructor Documentation	542
6.83.2.1	ActiveMQTempDestinationMarshaller	542
6.83.2.2	~ActiveMQTempDestinationMarshaller	542
6.83.3	Member Function Documentation	542
6.83.3.1	looseMarshal	542
6.83.3.2	looseUnmarshal	543
6.83.3.3	tightMarshal1	543
6.83.3.4	tightMarshal2	544
6.83.3.5	tightUnmarshal	544
6.84	activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller Class Reference	545
6.84.1	Detailed Description	546
6.84.2	Constructor & Destructor Documentation	546
6.84.2.1	ActiveMQTempDestinationMarshaller	546

6.84.2.2	<code>~ActiveMQTempDestinationMarshaller</code>	546
6.84.3	Member Function Documentation	546
6.84.3.1	<code>looseMarshal</code>	546
6.84.3.2	<code>looseUnmarshal</code>	547
6.84.3.3	<code>tightMarshal1</code>	547
6.84.3.4	<code>tightMarshal2</code>	548
6.84.3.5	<code>tightUnmarshal</code>	548
6.85	<code>activemq::commands::ActiveMQTempQueue</code> Class Reference	549
6.85.1	Constructor & Destructor Documentation	550
6.85.1.1	<code>ActiveMQTempQueue</code>	550
6.85.1.2	<code>ActiveMQTempQueue</code>	550
6.85.1.3	<code>~ActiveMQTempQueue</code>	550
6.85.2	Member Function Documentation	550
6.85.2.1	<code>clone</code>	550
6.85.2.2	<code>cloneDataStructure</code>	550
6.85.2.3	<code>copy</code>	550
6.85.2.4	<code>copyDataStructure</code>	550
6.85.2.5	<code>destroy</code>	551
6.85.2.6	<code>equals</code>	551
6.85.2.7	<code>getCMSDestination</code>	551
6.85.2.8	<code>getCMSProperties</code>	551
6.85.2.9	<code>getDataStructureType</code>	552
6.85.2.10	<code>getDestinationType</code>	552
6.85.2.11	<code>getQueueName</code>	552
6.85.2.12	<code>toString</code>	552
6.85.3	Field Documentation	552
6.85.3.1	<code>ID_ACTIVEMQTEMPQUEUE</code>	552
6.86	<code>activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller</code> Class Reference	552
6.86.1	Detailed Description	553
6.86.2	Constructor & Destructor Documentation	554
6.86.2.1	<code>ActiveMQTempQueueMarshaller</code>	554
6.86.2.2	<code>~ActiveMQTempQueueMarshaller</code>	554
6.86.3	Member Function Documentation	554
6.86.3.1	<code>createObject</code>	554
6.86.3.2	<code>getDataStructureType</code>	554
6.86.3.3	<code>looseMarshal</code>	554

6.86.3.4	looseUnmarshal	555
6.86.3.5	tightMarshal1	555
6.86.3.6	tightMarshal2	555
6.86.3.7	tightUnmarshal	556
6.87	activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	
	Class Reference	556
6.87.1	Detailed Description	557
6.87.2	Constructor & Destructor Documentation	558
6.87.2.1	ActiveMQTempQueueMarshaller	558
6.87.2.2	~ActiveMQTempQueueMarshaller	558
6.87.3	Member Function Documentation	558
6.87.3.1	createObject	558
6.87.3.2	getDataStructureType	558
6.87.3.3	looseMarshal	558
6.87.3.4	looseUnmarshal	559
6.87.3.5	tightMarshal1	559
6.87.3.6	tightMarshal2	559
6.87.3.7	tightUnmarshal	560
6.88	activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	
	Class Reference	560
6.88.1	Detailed Description	561
6.88.2	Constructor & Destructor Documentation	562
6.88.2.1	ActiveMQTempQueueMarshaller	562
6.88.2.2	~ActiveMQTempQueueMarshaller	562
6.88.3	Member Function Documentation	562
6.88.3.1	createObject	562
6.88.3.2	getDataStructureType	562
6.88.3.3	looseMarshal	562
6.88.3.4	looseUnmarshal	563
6.88.3.5	tightMarshal1	563
6.88.3.6	tightMarshal2	563
6.88.3.7	tightUnmarshal	564
6.89	activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	
	Class Reference	564
6.89.1	Detailed Description	565
6.89.2	Constructor & Destructor Documentation	566
6.89.2.1	ActiveMQTempQueueMarshaller	566

6.89.2.2	~ActiveMQTempQueueMarshaller	566
6.89.3	Member Function Documentation	566
6.89.3.1	createObject	566
6.89.3.2	getDataStructureType	566
6.89.3.3	looseMarshal	566
6.89.3.4	looseUnmarshal	567
6.89.3.5	tightMarshal1	567
6.89.3.6	tightMarshal2	567
6.89.3.7	tightUnmarshal	568
6.90	activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller Class Reference	568
6.90.1	Detailed Description	569
6.90.2	Constructor & Destructor Documentation	570
6.90.2.1	ActiveMQTempQueueMarshaller	570
6.90.2.2	~ActiveMQTempQueueMarshaller	570
6.90.3	Member Function Documentation	570
6.90.3.1	createObject	570
6.90.3.2	getDataStructureType	570
6.90.3.3	looseMarshal	570
6.90.3.4	looseUnmarshal	571
6.90.3.5	tightMarshal1	571
6.90.3.6	tightMarshal2	571
6.90.3.7	tightUnmarshal	572
6.91	activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller Class Reference	572
6.91.1	Detailed Description	573
6.91.2	Constructor & Destructor Documentation	574
6.91.2.1	ActiveMQTempQueueMarshaller	574
6.91.2.2	~ActiveMQTempQueueMarshaller	574
6.91.3	Member Function Documentation	574
6.91.3.1	createObject	574
6.91.3.2	getDataStructureType	574
6.91.3.3	looseMarshal	574
6.91.3.4	looseUnmarshal	575
6.91.3.5	tightMarshal1	575
6.91.3.6	tightMarshal2	575
6.91.3.7	tightUnmarshal	576

6.92	activemq::commands::ActiveMQTempTopic Class Reference	576
6.92.1	Constructor & Destructor Documentation	578
6.92.1.1	ActiveMQTempTopic	578
6.92.1.2	ActiveMQTempTopic	578
6.92.1.3	~ActiveMQTempTopic	578
6.92.2	Member Function Documentation	578
6.92.2.1	clone	578
6.92.2.2	cloneDataStructure	578
6.92.2.3	copy	578
6.92.2.4	copyDataStructure	578
6.92.2.5	destroy	579
6.92.2.6	equals	579
6.92.2.7	getCMSDestination	579
6.92.2.8	getCMSProperties	579
6.92.2.9	getDataStructureType	580
6.92.2.10	getDestinationType	580
6.92.2.11	getTopicName	580
6.92.2.12	toString	580
6.92.3	Field Documentation	580
6.92.3.1	ID_ACTIVEMQTEMPTOPIC	580
6.93	activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller Class Reference	580
6.93.1	Detailed Description	581
6.93.2	Constructor & Destructor Documentation	582
6.93.2.1	ActiveMQTempTopicMarshaller	582
6.93.2.2	~ActiveMQTempTopicMarshaller	582
6.93.3	Member Function Documentation	582
6.93.3.1	createObject	582
6.93.3.2	getDataStructureType	582
6.93.3.3	looseMarshal	582
6.93.3.4	looseUnmarshal	583
6.93.3.5	tightMarshal1	583
6.93.3.6	tightMarshal2	583
6.93.3.7	tightUnmarshal	584
6.94	activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller Class Reference	584
6.94.1	Detailed Description	585

6.94.2	Constructor & Destructor Documentation	586
6.94.2.1	ActiveMQTempTopicMarshaller	586
6.94.2.2	~ActiveMQTempTopicMarshaller	586
6.94.3	Member Function Documentation	586
6.94.3.1	createObject	586
6.94.3.2	getDataStructureType	586
6.94.3.3	looseMarshal	586
6.94.3.4	looseUnmarshal	587
6.94.3.5	tightMarshal1	587
6.94.3.6	tightMarshal2	587
6.94.3.7	tightUnmarshal	588
6.95	activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	
	Class Reference	588
6.95.1	Detailed Description	589
6.95.2	Constructor & Destructor Documentation	590
6.95.2.1	ActiveMQTempTopicMarshaller	590
6.95.2.2	~ActiveMQTempTopicMarshaller	590
6.95.3	Member Function Documentation	590
6.95.3.1	createObject	590
6.95.3.2	getDataStructureType	590
6.95.3.3	looseMarshal	590
6.95.3.4	looseUnmarshal	591
6.95.3.5	tightMarshal1	591
6.95.3.6	tightMarshal2	591
6.95.3.7	tightUnmarshal	592
6.96	activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	
	Class Reference	592
6.96.1	Detailed Description	593
6.96.2	Constructor & Destructor Documentation	594
6.96.2.1	ActiveMQTempTopicMarshaller	594
6.96.2.2	~ActiveMQTempTopicMarshaller	594
6.96.3	Member Function Documentation	594
6.96.3.1	createObject	594
6.96.3.2	getDataStructureType	594
6.96.3.3	looseMarshal	594
6.96.3.4	looseUnmarshal	595
6.96.3.5	tightMarshal1	595

6.96.3.6	tightMarshal2	595
6.96.3.7	tightUnmarshal	596
6.97	activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	
	Class Reference	596
6.97.1	Detailed Description	597
6.97.2	Constructor & Destructor Documentation	598
6.97.2.1	ActiveMQTempTopicMarshaller	598
6.97.2.2	~ActiveMQTempTopicMarshaller	598
6.97.3	Member Function Documentation	598
6.97.3.1	createObject	598
6.97.3.2	getDataStructureType	598
6.97.3.3	looseMarshal	598
6.97.3.4	looseUnmarshal	599
6.97.3.5	tightMarshal1	599
6.97.3.6	tightMarshal2	599
6.97.3.7	tightUnmarshal	600
6.98	activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	
	Class Reference	600
6.98.1	Detailed Description	601
6.98.2	Constructor & Destructor Documentation	602
6.98.2.1	ActiveMQTempTopicMarshaller	602
6.98.2.2	~ActiveMQTempTopicMarshaller	602
6.98.3	Member Function Documentation	602
6.98.3.1	createObject	602
6.98.3.2	getDataStructureType	602
6.98.3.3	looseMarshal	602
6.98.3.4	looseUnmarshal	603
6.98.3.5	tightMarshal1	603
6.98.3.6	tightMarshal2	603
6.98.3.7	tightUnmarshal	604
6.99	activemq::commands::ActiveMQTextMessage Class Reference	604
6.99.1	Constructor & Destructor Documentation	606
6.99.1.1	ActiveMQTextMessage	606
6.99.1.2	~ActiveMQTextMessage	606
6.99.2	Member Function Documentation	606
6.99.2.1	beforeMarshal	606
6.99.2.2	clearBody	606

6.99.2.3	clone	606
6.99.2.4	cloneDataStructure	607
6.99.2.5	copyDataStructure	607
6.99.2.6	equals	607
6.99.2.7	getDataStructureType	607
6.99.2.8	getSize	608
6.99.2.9	getText	608
6.99.2.10	setText	608
6.99.2.11	setText	608
6.99.2.12	toString	609
6.99.3	Field Documentation	609
6.99.3.1	ID_ACTIVEMQTEXTMESSAGE	609
6.99.3.2	text	609
6.100	activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	
	Class Reference	609
6.100.1	Detailed Description	610
6.100.2	Constructor & Destructor Documentation	610
6.100.2.1	ActiveMQTextMessageMarshaller	610
6.100.2.2	~ActiveMQTextMessageMarshaller	610
6.100.3	Member Function Documentation	610
6.100.3.1	createObject	610
6.100.3.2	getDataStructureType	611
6.100.3.3	looseMarshal	611
6.100.3.4	looseUnmarshal	611
6.100.3.5	tightMarshal1	612
6.100.3.6	tightMarshal2	612
6.100.3.7	tightUnmarshal	612
6.101	activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	
	Class Reference	613
6.101.1	Detailed Description	614
6.101.2	Constructor & Destructor Documentation	614
6.101.2.1	ActiveMQTextMessageMarshaller	614
6.101.2.2	~ActiveMQTextMessageMarshaller	614
6.101.3	Member Function Documentation	614
6.101.3.1	createObject	614
6.101.3.2	getDataStructureType	614
6.101.3.3	looseMarshal	615

6.101.3.4 looseUnmarshal	615
6.101.3.5 tightMarshal1	615
6.101.3.6 tightMarshal2	616
6.101.3.7 tightUnmarshal	616
6.102activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	
Class Reference	617
6.102.1 Detailed Description	618
6.102.2 Constructor & Destructor Documentation	618
6.102.2.1 ActiveMQTextMessageMarshaller	618
6.102.2.2 ~ActiveMQTextMessageMarshaller	618
6.102.3 Member Function Documentation	618
6.102.3.1 createObject	618
6.102.3.2 getDataStructureType	618
6.102.3.3 looseMarshal	619
6.102.3.4 looseUnmarshal	619
6.102.3.5 tightMarshal1	619
6.102.3.6 tightMarshal2	620
6.102.3.7 tightUnmarshal	620
6.103activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	
Class Reference	621
6.103.1 Detailed Description	622
6.103.2 Constructor & Destructor Documentation	622
6.103.2.1 ActiveMQTextMessageMarshaller	622
6.103.2.2 ~ActiveMQTextMessageMarshaller	622
6.103.3 Member Function Documentation	622
6.103.3.1 createObject	622
6.103.3.2 getDataStructureType	622
6.103.3.3 looseMarshal	623
6.103.3.4 looseUnmarshal	623
6.103.3.5 tightMarshal1	623
6.103.3.6 tightMarshal2	624
6.103.3.7 tightUnmarshal	624
6.104activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	
Class Reference	625
6.104.1 Detailed Description	626
6.104.2 Constructor & Destructor Documentation	626
6.104.2.1 ActiveMQTextMessageMarshaller	626

6.104.2.2 ~ActiveMQTextMessageMarshaller	626
6.104.3 Member Function Documentation	626
6.104.3.1 createObject	626
6.104.3.2 getDataStructureType	626
6.104.3.3 looseMarshal	627
6.104.3.4 looseUnmarshal	627
6.104.3.5 tightMarshal1	627
6.104.3.6 tightMarshal2	628
6.104.3.7 tightUnmarshal	628
6.105activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	
Class Reference	629
6.105.1 Detailed Description	630
6.105.2 Constructor & Destructor Documentation	630
6.105.2.1 ActiveMQTextMessageMarshaller	630
6.105.2.2 ~ActiveMQTextMessageMarshaller	630
6.105.3 Member Function Documentation	630
6.105.3.1 createObject	630
6.105.3.2 getDataStructureType	630
6.105.3.3 looseMarshal	631
6.105.3.4 looseUnmarshal	631
6.105.3.5 tightMarshal1	631
6.105.3.6 tightMarshal2	632
6.105.3.7 tightUnmarshal	632
6.106activemq::commands::ActiveMQTopic Class Reference	633
6.106.1 Constructor & Destructor Documentation	634
6.106.1.1 ActiveMQTopic	634
6.106.1.2 ActiveMQTopic	634
6.106.1.3 ~ActiveMQTopic	634
6.106.2 Member Function Documentation	634
6.106.2.1 clone	634
6.106.2.2 cloneDataStructure	634
6.106.2.3 copy	634
6.106.2.4 copyDataStructure	635
6.106.2.5 equals	635
6.106.2.6 getCMSDestination	635
6.106.2.7 getCMSProperties	635
6.106.2.8 getDataStructureType	636

6.106.2.9	getDestinationType	636
6.106.2.10	getTopicName	636
6.106.2.11	toString	636
6.106.3	Field Documentation	636
6.106.3.1	ID_ACTIVEMQTOPIC	636
6.107	activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller Class Reference	636
6.107.1	Detailed Description	637
6.107.2	Constructor & Destructor Documentation	638
6.107.2.1	ActiveMQTopicMarshaller	638
6.107.2.2	~ActiveMQTopicMarshaller	638
6.107.3	Member Function Documentation	638
6.107.3.1	createObject	638
6.107.3.2	getDataStructureType	638
6.107.3.3	looseMarshal	638
6.107.3.4	looseUnmarshal	639
6.107.3.5	tightMarshal1	639
6.107.3.6	tightMarshal2	639
6.107.3.7	tightUnmarshal	640
6.108	activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller Class Reference	640
6.108.1	Detailed Description	641
6.108.2	Constructor & Destructor Documentation	642
6.108.2.1	ActiveMQTopicMarshaller	642
6.108.2.2	~ActiveMQTopicMarshaller	642
6.108.3	Member Function Documentation	642
6.108.3.1	createObject	642
6.108.3.2	getDataStructureType	642
6.108.3.3	looseMarshal	642
6.108.3.4	looseUnmarshal	643
6.108.3.5	tightMarshal1	643
6.108.3.6	tightMarshal2	643
6.108.3.7	tightUnmarshal	644
6.109	activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller Class Reference	644
6.109.1	Detailed Description	645
6.109.2	Constructor & Destructor Documentation	646

6.109.2.1 ActiveMQTopicMarshaller	646
6.109.2.2 ~ActiveMQTopicMarshaller	646
6.109.3 Member Function Documentation	646
6.109.3.1 createObject	646
6.109.3.2 getDataStructureType	646
6.109.3.3 looseMarshal	646
6.109.3.4 looseUnmarshal	647
6.109.3.5 tightMarshal1	647
6.109.3.6 tightMarshal2	647
6.109.3.7 tightUnmarshal	648
6.110activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller Class Reference	648
6.110.1 Detailed Description	649
6.110.2 Constructor & Destructor Documentation	650
6.110.2.1 ActiveMQTopicMarshaller	650
6.110.2.2 ~ActiveMQTopicMarshaller	650
6.110.3 Member Function Documentation	650
6.110.3.1 createObject	650
6.110.3.2 getDataStructureType	650
6.110.3.3 looseMarshal	650
6.110.3.4 looseUnmarshal	651
6.110.3.5 tightMarshal1	651
6.110.3.6 tightMarshal2	651
6.110.3.7 tightUnmarshal	652
6.111activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller Class Reference	652
6.111.1 Detailed Description	653
6.111.2 Constructor & Destructor Documentation	654
6.111.2.1 ActiveMQTopicMarshaller	654
6.111.2.2 ~ActiveMQTopicMarshaller	654
6.111.3 Member Function Documentation	654
6.111.3.1 createObject	654
6.111.3.2 getDataStructureType	654
6.111.3.3 looseMarshal	654
6.111.3.4 looseUnmarshal	655
6.111.3.5 tightMarshal1	655
6.111.3.6 tightMarshal2	655

6.111.3.7	tightUnmarshal	656
6.112	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller Class Reference	656
6.112.1	Detailed Description	657
6.112.2	Constructor & Destructor Documentation	658
6.112.2.1	ActiveMQTopicMarshaller	658
6.112.2.2	~ActiveMQTopicMarshaller	658
6.112.3	Member Function Documentation	658
6.112.3.1	createObject	658
6.112.3.2	getDataStructureType	658
6.112.3.3	looseMarshal	658
6.112.3.4	looseUnmarshal	659
6.112.3.5	tightMarshal1	659
6.112.3.6	tightMarshal2	659
6.112.3.7	tightUnmarshal	660
6.113	activemq::core::ActiveMQTransactionContext Class Reference	660
6.113.1	Detailed Description	661
6.113.2	Constructor & Destructor Documentation	661
6.113.2.1	ActiveMQTransactionContext	661
6.113.2.2	~ActiveMQTransactionContext	662
6.113.3	Member Function Documentation	662
6.113.3.1	addSynchronization	662
6.113.3.2	begin	662
6.113.3.3	commit	662
6.113.3.4	getTransactionId	662
6.113.3.5	isInTransaction	663
6.113.3.6	removeSynchronization	663
6.113.3.7	rollback	663
6.114	decaf::util::zip::Adler32 Class Reference	663
6.114.1	Detailed Description	664
6.114.2	Constructor & Destructor Documentation	664
6.114.2.1	Adler32	664
6.114.2.2	~Adler32	664
6.114.3	Member Function Documentation	664
6.114.3.1	getValue	664
6.114.3.2	reset	664
6.114.3.3	update	665

6.114.3.4 update	665
6.114.3.5 update	665
6.114.3.6 update	666
6.115decaf::lang::Appendable Class Reference	666
6.115.1 Detailed Description	666
6.115.2 Constructor & Destructor Documentation	667
6.115.2.1 ~Appendable	667
6.115.3 Member Function Documentation	667
6.115.3.1 append	667
6.115.3.2 append	667
6.115.3.3 append	668
6.116decaf::internal::AprPool Class Reference	668
6.116.1 Detailed Description	668
6.116.2 Constructor & Destructor Documentation	669
6.116.2.1 AprPool	669
6.116.2.2 ~AprPool	669
6.116.3 Member Function Documentation	669
6.116.3.1 cleanup	669
6.116.3.2 getAprPool	669
6.116.3.3 getGlobalPool	669
6.117decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference	669
6.117.1 Detailed Description	671
6.117.2 Member Typedef Documentation	672
6.117.2.1 ConstReferenceType	672
6.117.2.2 CounterType	672
6.117.2.3 PointerType	672
6.117.2.4 ReferenceType	672
6.117.3 Constructor & Destructor Documentation	672
6.117.3.1 ArrayPointer	672
6.117.3.2 ArrayPointer	672
6.117.3.3 ArrayPointer	672
6.117.3.4 ArrayPointer	673
6.117.3.5 ~ArrayPointer	673
6.117.4 Member Function Documentation	673
6.117.4.1 clone	673
6.117.4.2 get	673

6.117.4.3	length	674
6.117.4.4	operator!	674
6.117.4.5	operator!=	674
6.117.4.6	operator=	674
6.117.4.7	operator=	674
6.117.4.8	operator==	674
6.117.4.9	operator[]	674
6.117.4.10	operator[]	675
6.117.4.11	release	675
6.117.4.12	reset	675
6.117.4.13	swap	675
6.117.5	Friends And Related Function Documentation	676
6.117.5.1	operator!=	676
6.117.5.2	operator!=	676
6.117.5.3	operator==	676
6.117.5.4	operator==	676
6.118	decaf::lang::ArrayPointerComparator< T, R > Class Template Reference	676
6.118.1	Detailed Description	677
6.118.2	Member Function Documentation	677
6.118.2.1	compare	677
6.118.2.2	operator()	677
6.119	decaf::util::concurrent::atomic::AtomicBoolean Class Reference	677
6.119.1	Detailed Description	678
6.119.2	Constructor & Destructor Documentation	678
6.119.2.1	AtomicBoolean	678
6.119.2.2	AtomicBoolean	678
6.119.2.3	~AtomicBoolean	678
6.119.3	Member Function Documentation	678
6.119.3.1	compareAndSet	678
6.119.3.2	get	679
6.119.3.3	getAndSet	679
6.119.3.4	set	679
6.119.3.5	toString	679
6.120	decaf::util::concurrent::atomic::AtomicInteger Class Reference	680
6.120.1	Detailed Description	681
6.120.2	Constructor & Destructor Documentation	681

6.120.2.1 AtomicInteger	681
6.120.2.2 AtomicInteger	681
6.120.2.3 ~AtomicInteger	681
6.120.3 Member Function Documentation	681
6.120.3.1 addAndGet	681
6.120.3.2 compareAndSet	682
6.120.3.3 decrementAndGet	682
6.120.3.4 doubleValue	682
6.120.3.5 floatValue	682
6.120.3.6 get	683
6.120.3.7 getAndAdd	683
6.120.3.8 getAndDecrement	683
6.120.3.9 getAndIncrement	683
6.120.3.10 getAndSet	683
6.120.3.11 incrementAndGet	684
6.120.3.12 intValue	684
6.120.3.13 longValue	684
6.120.3.14 set	684
6.120.3.15 toString	685
6.121 decaf::util::concurrent::atomic::AtomicRefCounter Class Reference	685
6.121.1 Constructor & Destructor Documentation	686
6.121.1.1 AtomicRefCounter	686
6.121.1.2 AtomicRefCounter	686
6.121.1.3 ~AtomicRefCounter	686
6.121.2 Member Function Documentation	686
6.121.2.1 release	686
6.121.2.2 swap	687
6.122 decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference	687
6.122.1 Detailed Description	688
6.122.2 Constructor & Destructor Documentation	688
6.122.2.1 AtomicReference	688
6.122.2.2 AtomicReference	688
6.122.2.3 ~AtomicReference	688
6.122.3 Member Function Documentation	688
6.122.3.1 compareAndSet	688
6.122.3.2 get	689

6.122.3.3	getAndSet	689
6.122.3.4	set	689
6.122.3.5	toString	689
6.123	activemq::transport::failover::BackupTransport Class Reference	690
6.123.1	Constructor & Destructor Documentation	690
6.123.1.1	BackupTransport	690
6.123.1.2	~BackupTransport	690
6.123.2	Member Function Documentation	690
6.123.2.1	getTransport	690
6.123.2.2	getUri	691
6.123.2.3	isClosed	691
6.123.2.4	onException	691
6.123.2.5	setClosed	691
6.123.2.6	setTransport	691
6.123.2.7	setUri	692
6.124	activemq::transport::failover::BackupTransportPool Class Reference	692
6.124.1	Constructor & Destructor Documentation	693
6.124.1.1	BackupTransportPool	693
6.124.1.2	BackupTransportPool	693
6.124.1.3	~BackupTransportPool	693
6.124.2	Member Function Documentation	693
6.124.2.1	getBackup	693
6.124.2.2	getBackupPoolSize	693
6.124.2.3	isEnabled	693
6.124.2.4	isPending	694
6.124.2.5	iterate	694
6.124.2.6	setBackupPoolSize	694
6.124.2.7	setEnabled	694
6.124.3	Friends And Related Function Documentation	694
6.124.3.1	BackupTransport	694
6.125	activemq::commands::BaseCommand Class Reference	694
6.125.1	Constructor & Destructor Documentation	696
6.125.1.1	BaseCommand	696
6.125.1.2	~BaseCommand	696
6.125.2	Member Function Documentation	696
6.125.2.1	copyDataStructure	696

6.125.2.2 equals	696
6.125.2.3 getCommandId	697
6.125.2.4 isBrokerInfo	698
6.125.2.5 isConnectionInfo	698
6.125.2.6 isConsumerInfo	698
6.125.2.7 isKeepAliveInfo	698
6.125.2.8 isMessage	698
6.125.2.9 isMessageAck	698
6.125.2.10sMessageDispatch	698
6.125.2.11sMessageDispatchNotification	699
6.125.2.12sProducerAck	699
6.125.2.13sProducerInfo	699
6.125.2.14sRemoveInfo	699
6.125.2.15sRemoveSubscriptionInfo	699
6.125.2.16sResponse	699
6.125.2.17sResponseRequired	699
6.125.2.18sShutdownInfo	700
6.125.2.19sTransactionInfo	700
6.125.2.20sWireFormatInfo	700
6.125.2.21setCommandId	700
6.125.2.22setResponseRequired	700
6.125.2.23oString	700
6.126activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller Class Reference	701
6.126.1 Detailed Description	702
6.126.2 Constructor & Destructor Documentation	702
6.126.2.1 BaseCommandMarshaller	702
6.126.2.2 ~BaseCommandMarshaller	702
6.126.3 Member Function Documentation	702
6.126.3.1 looseMarshal	702
6.126.3.2 looseUnmarshal	703
6.126.3.3 tightMarshal1	704
6.126.3.4 tightMarshal2	706
6.126.3.5 tightUnmarshal	707
6.127activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller Class Reference	708
6.127.1 Detailed Description	709

6.127.2 Constructor & Destructor Documentation	709
6.127.2.1 BaseCommandMarshaller	709
6.127.2.2 ~BaseCommandMarshaller	709
6.127.3 Member Function Documentation	709
6.127.3.1 looseMarshal	709
6.127.3.2 looseUnmarshal	710
6.127.3.3 tightMarshal1	711
6.127.3.4 tightMarshal2	712
6.127.3.5 tightUnmarshal	713
6.128activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller Class Reference	714
6.128.1 Detailed Description	715
6.128.2 Constructor & Destructor Documentation	716
6.128.2.1 BaseCommandMarshaller	716
6.128.2.2 ~BaseCommandMarshaller	716
6.128.3 Member Function Documentation	716
6.128.3.1 looseMarshal	716
6.128.3.2 looseUnmarshal	717
6.128.3.3 tightMarshal1	718
6.128.3.4 tightMarshal2	719
6.128.3.5 tightUnmarshal	720
6.129activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller Class Reference	721
6.129.1 Detailed Description	722
6.129.2 Constructor & Destructor Documentation	722
6.129.2.1 BaseCommandMarshaller	722
6.129.2.2 ~BaseCommandMarshaller	722
6.129.3 Member Function Documentation	722
6.129.3.1 looseMarshal	722
6.129.3.2 looseUnmarshal	723
6.129.3.3 tightMarshal1	724
6.129.3.4 tightMarshal2	726
6.129.3.5 tightUnmarshal	727
6.130activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller Class Reference	728
6.130.1 Detailed Description	729
6.130.2 Constructor & Destructor Documentation	729

6.130.2.1 BaseCommandMarshaller	729
6.130.2.2 ~BaseCommandMarshaller	729
6.130.3 Member Function Documentation	729
6.130.3.1 looseMarshal	729
6.130.3.2 looseUnmarshal	730
6.130.3.3 tightMarshal1	731
6.130.3.4 tightMarshal2	732
6.130.3.5 tightUnmarshal	733
6.131activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller Class Reference	734
6.131.1 Detailed Description	735
6.131.2 Constructor & Destructor Documentation	736
6.131.2.1 BaseCommandMarshaller	736
6.131.2.2 ~BaseCommandMarshaller	736
6.131.3 Member Function Documentation	736
6.131.3.1 looseMarshal	736
6.131.3.2 looseUnmarshal	737
6.131.3.3 tightMarshal1	738
6.131.3.4 tightMarshal2	739
6.131.3.5 tightUnmarshal	740
6.132activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller Class Reference	741
6.132.1 Detailed Description	748
6.132.2 Constructor & Destructor Documentation	748
6.132.2.1 ~BaseDataStreamMarshaller	748
6.132.3 Member Function Documentation	748
6.132.3.1 looseMarshal	748
6.132.3.2 looseMarshalBrokerError	748
6.132.3.3 looseMarshalCachedObject	749
6.132.3.4 looseMarshalLong	749
6.132.3.5 looseMarshalNestedObject	749
6.132.3.6 looseMarshalObjectArray	750
6.132.3.7 looseMarshalString	750
6.132.3.8 looseUnmarshal	750
6.132.3.9 looseUnmarshalBrokerError	751
6.132.3.10looseUnmarshalByteArray	751
6.132.3.11looseUnmarshalCachedObject	751

6.132.3.12ooseUnmarshalConstByteArray	752
6.132.3.13ooseUnmarshalLong	752
6.132.3.14ooseUnmarshalNestedObject	753
6.132.3.15ooseUnmarshalString	753
6.132.3.16readAsciiString	753
6.132.3.17ightMarshal1	754
6.132.3.18ightMarshal2	754
6.132.3.19ightMarshalBrokerError1	754
6.132.3.20ightMarshalBrokerError2	755
6.132.3.21ightMarshalCachedObject1	755
6.132.3.22ightMarshalCachedObject2	756
6.132.3.23ightMarshalLong1	756
6.132.3.24ightMarshalLong2	756
6.132.3.25ightMarshalNestedObject1	757
6.132.3.26ightMarshalNestedObject2	757
6.132.3.27ightMarshalObjectArray1	758
6.132.3.28ightMarshalObjectArray2	758
6.132.3.29ightMarshalString1	759
6.132.3.30ightMarshalString2	759
6.132.3.31ightUnmarshal	759
6.132.3.32ightUnmarshalBrokerError	760
6.132.3.33ightUnmarshalByteArray	760
6.132.3.34ightUnmarshalCachedObject	761
6.132.3.35ightUnmarshalConstByteArray	761
6.132.3.36ightUnmarshalLong	761
6.132.3.37ightUnmarshalNestedObject	762
6.132.3.38ightUnmarshalString	762
6.132.3.39oHexFromBytes	763
6.132.3.40oString	763
6.132.3.41toString	763
6.132.3.42oString	763
6.133activemq::commands::BaseDataStructure Class Reference	764
6.133.1 Constructor & Destructor Documentation	765
6.133.1.1 ~BaseDataStructure	765
6.133.2 Member Function Documentation	765
6.133.2.1 afterMarshal	765

6.133.2.2 afterUnmarshal	765
6.133.2.3 beforeMarshal	765
6.133.2.4 beforeUnmarshal	766
6.133.2.5 copyDataStructure	766
6.133.2.6 equals	766
6.133.2.7 getMarshaledForm	766
6.133.2.8 isMarshalAware	767
6.133.2.9 setMarshaledForm	767
6.133.2.10 toString	767
6.134binary_function Class Reference	768
6.135decaf::net::BindException Class Reference	768
6.135.1 Constructor & Destructor Documentation	769
6.135.1.1 BindException	769
6.135.1.2 BindException	769
6.135.1.3 BindException	769
6.135.1.4 BindException	770
6.135.1.5 BindException	770
6.135.1.6 BindException	770
6.135.1.7 ~BindException	770
6.135.2 Member Function Documentation	770
6.135.2.1 clone	770
6.136decaf::io::BlockingByteArrayInputStream Class Reference	771
6.136.1 Detailed Description	772
6.136.2 Constructor & Destructor Documentation	772
6.136.2.1 BlockingByteArrayInputStream	772
6.136.2.2 BlockingByteArrayInputStream	772
6.136.2.3 ~BlockingByteArrayInputStream	773
6.136.3 Member Function Documentation	773
6.136.3.1 available	773
6.136.3.2 close	773
6.136.3.3 doReadArrayBounded	773
6.136.3.4 doReadByte	773
6.136.3.5 setByteArray	774
6.136.3.6 skip	774
6.137decaf::util::concurrent::BlockingQueue< E > Class Template Reference	774
6.137.1 Detailed Description	775

6.137.2 Constructor & Destructor Documentation	777
6.137.2.1 ~BlockingQueue	777
6.137.3 Member Function Documentation	777
6.137.3.1 drainTo	777
6.137.3.2 drainTo	778
6.137.3.3 offer	779
6.137.3.4 poll	779
6.137.3.5 put	780
6.137.3.6 remainingCapacity	780
6.137.3.7 take	780
6.138decaf::lang::Boolean Class Reference	781
6.138.1 Constructor & Destructor Documentation	782
6.138.1.1 Boolean	782
6.138.1.2 Boolean	782
6.138.1.3 ~Boolean	782
6.138.2 Member Function Documentation	782
6.138.2.1 booleanValue	782
6.138.2.2 compareTo	782
6.138.2.3 compareTo	783
6.138.2.4 equals	783
6.138.2.5 equals	783
6.138.2.6 operator<	783
6.138.2.7 operator<	783
6.138.2.8 operator==	784
6.138.2.9 operator==	784
6.138.2.10parseBoolean	784
6.138.2.11toString	785
6.138.2.12toString	785
6.138.2.13valueOf	785
6.138.2.14valueOf	785
6.138.3 Field Documentation	785
6.138.3.1 _FALSE	785
6.138.3.2 _TRUE	785
6.139activemq::commands::BooleanExpression Class Reference	786
6.139.1 Constructor & Destructor Documentation	786
6.139.1.1 BooleanExpression	786

6.139.1.2 <code>~BooleanExpression</code>	786
6.139.2 Member Function Documentation	786
6.139.2.1 <code>cloneDataStructure</code>	786
6.139.2.2 <code>copyDataStructure</code>	787
6.139.2.3 <code>equals</code>	787
6.139.2.4 <code>toString</code>	787
6.140 <code>activemq::wireformat::openwire::utils::BooleanStream</code> Class Reference	787
6.140.1 Detailed Description	788
6.140.2 Constructor & Destructor Documentation	789
6.140.2.1 <code>BooleanStream</code>	789
6.140.2.2 <code>~BooleanStream</code>	789
6.140.3 Member Function Documentation	789
6.140.3.1 <code>clear</code>	789
6.140.3.2 <code>marshal</code>	789
6.140.3.3 <code>marshal</code>	789
6.140.3.4 <code>marshalledSize</code>	789
6.140.3.5 <code>readBoolean</code>	789
6.140.3.6 <code>unmarshal</code>	790
6.140.3.7 <code>writeBoolean</code>	790
6.141 <code>decaf::util::concurrent::BrokenBarrierException</code> Class Reference	790
6.141.1 Constructor & Destructor Documentation	791
6.141.1.1 <code>BrokenBarrierException</code>	791
6.141.1.2 <code>BrokenBarrierException</code>	791
6.141.1.3 <code>BrokenBarrierException</code>	791
6.141.1.4 <code>BrokenBarrierException</code>	791
6.141.1.5 <code>BrokenBarrierException</code>	792
6.141.1.6 <code>BrokenBarrierException</code>	792
6.141.1.7 <code>~BrokenBarrierException</code>	792
6.141.2 Member Function Documentation	792
6.141.2.1 <code>clone</code>	792
6.142 <code>activemq::commands::BrokerError</code> Class Reference	793
6.142.1 Detailed Description	794
6.142.2 Constructor & Destructor Documentation	794
6.142.2.1 <code>BrokerError</code>	794
6.142.2.2 <code>~BrokerError</code>	794
6.142.3 Member Function Documentation	794

6.142.3.1 cloneDataStructure	794
6.142.3.2 copyDataStructure	794
6.142.3.3 getCause	795
6.142.3.4 getDataStructureType	795
6.142.3.5 getExceptionClass	795
6.142.3.6 getMessage	795
6.142.3.7 getStackTraceElements	795
6.142.3.8 setCause	796
6.142.3.9 setExceptionClass	796
6.142.3.10 setMessage	796
6.142.3.11 setStackTraceElements	796
6.142.3.12 visit	796
6.143activemq::exceptions::BrokerException Class Reference	797
6.143.1 Constructor & Destructor Documentation	797
6.143.1.1 BrokerException	797
6.143.1.2 BrokerException	797
6.143.1.3 BrokerException	797
6.143.1.4 BrokerException	797
6.143.1.5 BrokerException	797
6.143.1.6 ~BrokerException	798
6.143.2 Member Function Documentation	798
6.143.2.1 clone	798
6.144activemq::commands::BrokerId Class Reference	798
6.144.1 Member Typedef Documentation	799
6.144.1.1 COMPARATOR	799
6.144.2 Constructor & Destructor Documentation	799
6.144.2.1 BrokerId	799
6.144.2.2 BrokerId	799
6.144.2.3 ~BrokerId	799
6.144.3 Member Function Documentation	799
6.144.3.1 cloneDataStructure	799
6.144.3.2 compareTo	799
6.144.3.3 copyDataStructure	799
6.144.3.4 equals	800
6.144.3.5 equals	800
6.144.3.6 getDataStructureType	800

6.144.3.7	getValue	800
6.144.3.8	getValue	800
6.144.3.9	operator<	800
6.144.3.10	operator=	800
6.144.3.11	operator==	800
6.144.3.12	setValue	800
6.144.3.13	toString	800
6.144.4	Field Documentation	801
6.144.4.1	ID_BROKERID	801
6.144.4.2	value	801
6.145	activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference	801
6.145.1	Detailed Description	802
6.145.2	Constructor & Destructor Documentation	802
6.145.2.1	BrokerIdMarshaller	802
6.145.2.2	~BrokerIdMarshaller	802
6.145.3	Member Function Documentation	802
6.145.3.1	createObject	802
6.145.3.2	getDataStructureType	802
6.145.3.3	looseMarshal	803
6.145.3.4	looseUnmarshal	803
6.145.3.5	tightMarshal1	803
6.145.3.6	tightMarshal2	804
6.145.3.7	tightUnmarshal	804
6.146	activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller Class Reference	805
6.146.1	Detailed Description	806
6.146.2	Constructor & Destructor Documentation	806
6.146.2.1	BrokerIdMarshaller	806
6.146.2.2	~BrokerIdMarshaller	806
6.146.3	Member Function Documentation	806
6.146.3.1	createObject	806
6.146.3.2	getDataStructureType	806
6.146.3.3	looseMarshal	806
6.146.3.4	looseUnmarshal	807
6.146.3.5	tightMarshal1	807
6.146.3.6	tightMarshal2	808
6.146.3.7	tightUnmarshal	808

6.147	activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference	808
6.147.1	Detailed Description	809
6.147.2	Constructor & Destructor Documentation	810
6.147.2.1	BrokerIdMarshaller	810
6.147.2.2	~BrokerIdMarshaller	810
6.147.3	Member Function Documentation	810
6.147.3.1	createObject	810
6.147.3.2	getDataStructureType	810
6.147.3.3	looseMarshal	810
6.147.3.4	looseUnmarshal	811
6.147.3.5	tightMarshal1	811
6.147.3.6	tightMarshal2	811
6.147.3.7	tightUnmarshal	812
6.148	activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller Class Reference	812
6.148.1	Detailed Description	813
6.148.2	Constructor & Destructor Documentation	814
6.148.2.1	BrokerIdMarshaller	814
6.148.2.2	~BrokerIdMarshaller	814
6.148.3	Member Function Documentation	814
6.148.3.1	createObject	814
6.148.3.2	getDataStructureType	814
6.148.3.3	looseMarshal	814
6.148.3.4	looseUnmarshal	815
6.148.3.5	tightMarshal1	815
6.148.3.6	tightMarshal2	815
6.148.3.7	tightUnmarshal	816
6.149	activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller Class Reference	816
6.149.1	Detailed Description	817
6.149.2	Constructor & Destructor Documentation	818
6.149.2.1	BrokerIdMarshaller	818
6.149.2.2	~BrokerIdMarshaller	818
6.149.3	Member Function Documentation	818
6.149.3.1	createObject	818
6.149.3.2	getDataStructureType	818
6.149.3.3	looseMarshal	818
6.149.3.4	looseUnmarshal	819

6.149.3.5 tightMarshal1	819
6.149.3.6 tightMarshal2	819
6.149.3.7 tightUnmarshal	820
6.150activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller Class Reference	820
6.150.1 Detailed Description	821
6.150.2 Constructor & Destructor Documentation	822
6.150.2.1 BrokerIdMarshaller	822
6.150.2.2 ~BrokerIdMarshaller	822
6.150.3 Member Function Documentation	822
6.150.3.1 createObject	822
6.150.3.2 getDataStructureType	822
6.150.3.3 looseMarshal	822
6.150.3.4 looseUnmarshal	823
6.150.3.5 tightMarshal1	823
6.150.3.6 tightMarshal2	823
6.150.3.7 tightUnmarshal	824
6.151activemq::commands::BrokerInfo Class Reference	824
6.151.1 Constructor & Destructor Documentation	826
6.151.1.1 BrokerInfo	826
6.151.1.2 ~BrokerInfo	826
6.151.2 Member Function Documentation	826
6.151.2.1 cloneDataStructure	826
6.151.2.2 copyDataStructure	826
6.151.2.3 equals	827
6.151.2.4 getBrokerId	827
6.151.2.5 getBrokerId	827
6.151.2.6 getBrokerName	827
6.151.2.7 getBrokerName	827
6.151.2.8 getBrokerUploadUrl	827
6.151.2.9 getBrokerUploadUrl	827
6.151.2.10getBrokerURL	827
6.151.2.11getBrokerURL	827
6.151.2.12getConnectionId	827
6.151.2.13getDataStructureType	827
6.151.2.14getNetworkProperties	828
6.151.2.15getNetworkProperties	828

6.151.2.16	getPeerBrokerInfos	828
6.151.2.17	getPeerBrokerInfos	828
6.151.2.18	sBrokerInfo	828
6.151.2.19	sDuplexConnection	829
6.151.2.20	sFaultTolerantConfiguration	829
6.151.2.21	isMasterBroker	829
6.151.2.22	sNetworkConnection	829
6.151.2.23	sSlaveBroker	829
6.151.2.24	setBrokerId	829
6.151.2.25	setBrokerName	829
6.151.2.26	setBrokerUploadUrl	829
6.151.2.27	setBrokerURL	829
6.151.2.28	setConnectionId	829
6.151.2.29	setDuplexConnection	829
6.151.2.30	setFaultTolerantConfiguration	829
6.151.2.31	setMasterBroker	829
6.151.2.32	setNetworkConnection	829
6.151.2.33	setNetworkProperties	829
6.151.2.34	setPeerBrokerInfos	829
6.151.2.35	setSlaveBroker	829
6.151.2.36	toString	829
6.151.2.37	visit	830
6.151.3	Field Documentation	831
6.151.3.1	brokerId	831
6.151.3.2	brokerName	831
6.151.3.3	brokerUploadUrl	831
6.151.3.4	brokerURL	831
6.151.3.5	connectionId	831
6.151.3.6	duplexConnection	831
6.151.3.7	faultTolerantConfiguration	831
6.151.3.8	ID_BROKERINFO	831
6.151.3.9	masterBroker	831
6.151.3.10	networkConnection	831
6.151.3.11	networkProperties	831
6.151.3.12	peerBrokerInfos	831
6.151.3.13	slaveBroker	831

6.152activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller Class Reference	831
6.152.1 Detailed Description	832
6.152.2 Constructor & Destructor Documentation	833
6.152.2.1 BrokerInfoMarshaller	833
6.152.2.2 ~BrokerInfoMarshaller	833
6.152.3 Member Function Documentation	833
6.152.3.1 createObject	833
6.152.3.2 getDataStructureType	833
6.152.3.3 looseMarshal	833
6.152.3.4 looseUnmarshal	834
6.152.3.5 tightMarshal1	834
6.152.3.6 tightMarshal2	834
6.152.3.7 tightUnmarshal	835
6.153activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller Class Reference	835
6.153.1 Detailed Description	836
6.153.2 Constructor & Destructor Documentation	837
6.153.2.1 BrokerInfoMarshaller	837
6.153.2.2 ~BrokerInfoMarshaller	837
6.153.3 Member Function Documentation	837
6.153.3.1 createObject	837
6.153.3.2 getDataStructureType	837
6.153.3.3 looseMarshal	837
6.153.3.4 looseUnmarshal	838
6.153.3.5 tightMarshal1	838
6.153.3.6 tightMarshal2	838
6.153.3.7 tightUnmarshal	839
6.154activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller Class Reference	839
6.154.1 Detailed Description	840
6.154.2 Constructor & Destructor Documentation	841
6.154.2.1 BrokerInfoMarshaller	841
6.154.2.2 ~BrokerInfoMarshaller	841
6.154.3 Member Function Documentation	841
6.154.3.1 createObject	841
6.154.3.2 getDataStructureType	841
6.154.3.3 looseMarshal	841
6.154.3.4 looseUnmarshal	842

6.154.3.5 tightMarshal1	842
6.154.3.6 tightMarshal2	842
6.154.3.7 tightUnmarshal	843
6.155activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller Class Reference	843
6.155.1 Detailed Description	844
6.155.2 Constructor & Destructor Documentation	845
6.155.2.1 BrokerInfoMarshaller	845
6.155.2.2 ~BrokerInfoMarshaller	845
6.155.3 Member Function Documentation	845
6.155.3.1 createObject	845
6.155.3.2 getDataStructureType	845
6.155.3.3 looseMarshal	845
6.155.3.4 looseUnmarshal	846
6.155.3.5 tightMarshal1	846
6.155.3.6 tightMarshal2	846
6.155.3.7 tightUnmarshal	847
6.156activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller Class Reference	847
6.156.1 Detailed Description	848
6.156.2 Constructor & Destructor Documentation	849
6.156.2.1 BrokerInfoMarshaller	849
6.156.2.2 ~BrokerInfoMarshaller	849
6.156.3 Member Function Documentation	849
6.156.3.1 createObject	849
6.156.3.2 getDataStructureType	849
6.156.3.3 looseMarshal	849
6.156.3.4 looseUnmarshal	850
6.156.3.5 tightMarshal1	850
6.156.3.6 tightMarshal2	850
6.156.3.7 tightUnmarshal	851
6.157activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller Class Reference	851
6.157.1 Detailed Description	852
6.157.2 Constructor & Destructor Documentation	853
6.157.2.1 BrokerInfoMarshaller	853
6.157.2.2 ~BrokerInfoMarshaller	853
6.157.3 Member Function Documentation	853
6.157.3.1 createObject	853

6.157.3.2	getDataStructureType	853
6.157.3.3	looseMarshal	853
6.157.3.4	looseUnmarshal	854
6.157.3.5	tightMarshal1	854
6.157.3.6	tightMarshal2	854
6.157.3.7	tightUnmarshal	855
6.158	decaf::nio::Buffer Class Reference	855
6.158.1	Detailed Description	857
6.158.2	Constructor & Destructor Documentation	858
6.158.2.1	Buffer	858
6.158.2.2	Buffer	858
6.158.2.3	~Buffer	858
6.158.3	Member Function Documentation	858
6.158.3.1	capacity	858
6.158.3.2	clear	858
6.158.3.3	flip	858
6.158.3.4	hasRemaining	859
6.158.3.5	isReadOnly	859
6.158.3.6	limit	859
6.158.3.7	limit	859
6.158.3.8	mark	860
6.158.3.9	position	860
6.158.3.10	position	860
6.158.3.11	remaining	860
6.158.3.12	reset	860
6.158.3.13	rewind	861
6.158.4	Field Documentation	861
6.158.4.1	_capacity	861
6.158.4.2	_limit	861
6.158.4.3	_mark	861
6.158.4.4	_markSet	861
6.158.4.5	_position	861
6.159	decaf::io::BufferedInputStream Class Reference	861
6.159.1	Detailed Description	863
6.159.2	Constructor & Destructor Documentation	863
6.159.2.1	BufferedInputStream	863

6.159.2.2	BufferedInputStream	864
6.159.2.3	~BufferedInputStream	864
6.159.3	Member Function Documentation	864
6.159.3.1	available	864
6.159.3.2	close	864
6.159.3.3	doReadArrayBounded	865
6.159.3.4	doReadByte	865
6.159.3.5	mark	865
6.159.3.6	markSupported	865
6.159.3.7	reset	865
6.159.3.8	skip	866
6.160	decaf::io::BufferedOutputStream Class Reference	867
6.160.1	Detailed Description	867
6.160.2	Constructor & Destructor Documentation	867
6.160.2.1	BufferedOutputStream	867
6.160.2.2	BufferedOutputStream	868
6.160.2.3	~BufferedOutputStream	868
6.160.3	Member Function Documentation	868
6.160.3.1	doWriteArray	868
6.160.3.2	doWriteArrayBounded	868
6.160.3.3	doWriteByte	868
6.160.3.4	flush	868
6.161	decaf::internal::nio::BufferFactory Class Reference	869
6.161.1	Detailed Description	871
6.161.2	Constructor & Destructor Documentation	871
6.161.2.1	~BufferFactory	871
6.161.3	Member Function Documentation	871
6.161.3.1	createByteBuffer	871
6.161.3.2	createByteBuffer	871
6.161.3.3	createByteBuffer	872
6.161.3.4	createCharBuffer	872
6.161.3.5	createCharBuffer	873
6.161.3.6	createCharBuffer	873
6.161.3.7	createDoubleBuffer	873
6.161.3.8	createDoubleBuffer	874
6.161.3.9	createDoubleBuffer	874

6.161.3.10	createFloatBuffer	875
6.161.3.11	createFloatBuffer	875
6.161.3.12	createFloatBuffer	876
6.161.3.13	createIntBuffer	876
6.161.3.14	createIntBuffer	876
6.161.3.15	createIntBuffer	877
6.161.3.16	createLongBuffer	877
6.161.3.17	createLongBuffer	878
6.161.3.18	createLongBuffer	878
6.161.3.19	createShortBuffer	878
6.161.3.20	createShortBuffer	879
6.161.3.21	createShortBuffer	879
6.162	decaf::nio::BufferOverflowException Class Reference	880
6.162.1	Constructor & Destructor Documentation	880
6.162.1.1	BufferOverflowException	880
6.162.1.2	BufferOverflowException	881
6.162.1.3	BufferOverflowException	881
6.162.1.4	BufferOverflowException	881
6.162.1.5	BufferOverflowException	881
6.162.1.6	BufferOverflowException	881
6.162.1.7	~BufferOverflowException	882
6.162.2	Member Function Documentation	882
6.162.2.1	clone	882
6.163	decaf::nio::BufferUnderflowException Class Reference	882
6.163.1	Constructor & Destructor Documentation	883
6.163.1.1	BufferUnderflowException	883
6.163.1.2	BufferUnderflowException	883
6.163.1.3	BufferUnderflowException	883
6.163.1.4	BufferUnderflowException	883
6.163.1.5	BufferUnderflowException	884
6.163.1.6	BufferUnderflowException	884
6.163.1.7	~BufferUnderflowException	884
6.163.2	Member Function Documentation	884
6.163.2.1	clone	884
6.164	decaf::lang::Byte Class Reference	884
6.164.1	Constructor & Destructor Documentation	886

6.164.1.1 Byte	886
6.164.1.2 Byte	887
6.164.1.3 ~Byte	887
6.164.2 Member Function Documentation	887
6.164.2.1 byteValue	887
6.164.2.2 compareTo	887
6.164.2.3 compareTo	887
6.164.2.4 decode	888
6.164.2.5 doubleValue	888
6.164.2.6 equals	888
6.164.2.7 equals	888
6.164.2.8 float Value	888
6.164.2.9 int Value	889
6.164.2.10 long Value	889
6.164.2.11 operator<	889
6.164.2.12 operator<	889
6.164.2.13 operator==	890
6.164.2.14 operator==	890
6.164.2.15 parseByte	890
6.164.2.16 parseByte	891
6.164.2.17 short Value	891
6.164.2.18 oString	891
6.164.2.19 oString	891
6.164.2.20 valueOf	892
6.164.2.21 valueOf	892
6.164.2.22 valueOf	892
6.164.3 Field Documentation	893
6.164.3.1 MAX_VALUE	893
6.164.3.2 MIN_VALUE	893
6.164.3.3 SIZE	893
6.165 decaf::internal::util::ByteArrayAdapter Class Reference	893
6.165.1 Detailed Description	897
6.165.2 Constructor & Destructor Documentation	898
6.165.2.1 ByteArrayAdapter	898
6.165.2.2 ByteArrayAdapter	898
6.165.2.3 ByteArrayAdapter	898

6.165.2.4 ByteArrayAdapter	899
6.165.2.5 ByteArrayAdapter	899
6.165.2.6 ByteArrayAdapter	899
6.165.2.7 ByteArrayAdapter	900
6.165.2.8 ByteArrayAdapter	900
6.165.2.9 ~ByteArrayAdapter	901
6.165.3 Member Function Documentation	901
6.165.3.1 clear	901
6.165.3.2 get	901
6.165.3.3 getByteArray	901
6.165.3.4 getCapacity	901
6.165.3.5 getChar	902
6.165.3.6 getCharArray	902
6.165.3.7 getCharCapacity	902
6.165.3.8 getDouble	902
6.165.3.9 getDoubleArray	903
6.165.3.10 getDoubleAt	903
6.165.3.11 getDoubleCapacity	903
6.165.3.12 getFloat	904
6.165.3.13 getFloatArray	904
6.165.3.14 getFloatAt	904
6.165.3.15 getFloatCapacity	905
6.165.3.16 getInt	905
6.165.3.17 getIntArray	905
6.165.3.18 getIntAt	905
6.165.3.19 getIntCapacity	906
6.165.3.20 getLong	906
6.165.3.21 getLongArray	906
6.165.3.22 getLongAt	907
6.165.3.23 getLongCapacity	907
6.165.3.24 getShort	907
6.165.3.25 getShortArray	908
6.165.3.26 getShortAt	908
6.165.3.27 getShortCapacity	908
6.165.3.28 operator[]	908
6.165.3.29 operator[]	909

6.165.3.30	put	909
6.165.3.31	putChar	909
6.165.3.32	putDouble	910
6.165.3.33	putDoubleAt	910
6.165.3.34	putFloat	910
6.165.3.35	putFloatAt	911
6.165.3.36	putInt	911
6.165.3.37	putIntAt	912
6.165.3.38	putLong	912
6.165.3.39	putLongAt	912
6.165.3.40	putShort	913
6.165.3.41	putShortAt	913
6.165.3.42	read	914
6.165.3.43	resize	914
6.165.3.44	write	914
6.166	decaf::internal::nio::ByteBuffer Class Reference	915
6.166.1	Detailed Description	925
6.166.2	Constructor & Destructor Documentation	926
6.166.2.1	ByteBuffer	926
6.166.2.2	ByteBuffer	927
6.166.2.3	ByteBuffer	927
6.166.2.4	ByteBuffer	927
6.166.2.5	~ByteBuffer	928
6.166.3	Member Function Documentation	928
6.166.3.1	array	928
6.166.3.2	arrayOffset	928
6.166.3.3	asCharBuffer	928
6.166.3.4	asDoubleBuffer	929
6.166.3.5	asFloatBuffer	929
6.166.3.6	asIntBuffer	930
6.166.3.7	asLongBuffer	930
6.166.3.8	asReadOnlyBuffer	930
6.166.3.9	asShortBuffer	931
6.166.3.10	compact	931
6.166.3.11	duplicate	931
6.166.3.12	get	932

6.166.3.13	get	932
6.166.3.14	getChar	933
6.166.3.15	getChar	933
6.166.3.16	getDouble	933
6.166.3.17	getDouble	934
6.166.3.18	getFloat	934
6.166.3.19	getFloat	934
6.166.3.20	getInt	935
6.166.3.21	getInt	935
6.166.3.22	getLong	935
6.166.3.23	getLong	936
6.166.3.24	getShort	936
6.166.3.25	getShort	936
6.166.3.26	hasArray	937
6.166.3.27	isReadOnly	937
6.166.3.28	put	937
6.166.3.29	put	937
6.166.3.30	putChar	938
6.166.3.31	putChar	938
6.166.3.32	putDouble	939
6.166.3.33	putDouble	939
6.166.3.34	putFloat	940
6.166.3.35	putFloat	940
6.166.3.36	putInt	940
6.166.3.37	putInt	941
6.166.3.38	putLong	941
6.166.3.39	putLong	942
6.166.3.40	putShort	942
6.166.3.41	putShort	943
6.166.3.42	setReadOnly	943
6.166.3.43	lice	943
6.167	decaf::io::ByteArrayInputStream Class Reference	944
6.167.1	Detailed Description	946
6.167.2	Constructor & Destructor Documentation	946
6.167.2.1	ByteArrayInputStream	946
6.167.2.2	ByteArrayInputStream	946

6.167.2.3	ByteArrayInputStream	947
6.167.2.4	ByteArrayInputStream	947
6.167.2.5	~ByteArrayInputStream	947
6.167.3	Member Function Documentation	947
6.167.3.1	available	947
6.167.3.2	doReadArrayBounded	948
6.167.3.3	doReadByte	948
6.167.3.4	mark	948
6.167.3.5	markSupported	948
6.167.3.6	reset	949
6.167.3.7	setByteArray	949
6.167.3.8	setByteArray	950
6.167.3.9	setByteArray	950
6.167.3.10	skip	950
6.168	decaf::io::ByteArrayOutputStream Class Reference	951
6.168.1	Constructor & Destructor Documentation	952
6.168.1.1	ByteArrayOutputStream	952
6.168.1.2	ByteArrayOutputStream	952
6.168.1.3	~ByteArrayOutputStream	952
6.168.2	Member Function Documentation	952
6.168.2.1	doWriteArrayBounded	952
6.168.2.2	doWriteByte	952
6.168.2.3	reset	952
6.168.2.4	size	953
6.168.2.5	toByteArray	953
6.168.2.6	toString	953
6.168.2.7	writeTo	953
6.169	decaf::nio::ByteBuffer Class Reference	954
6.169.1	Detailed Description	958
6.169.2	Constructor & Destructor Documentation	959
6.169.2.1	ByteBuffer	959
6.169.2.2	~ByteBuffer	959
6.169.3	Member Function Documentation	959
6.169.3.1	allocate	959
6.169.3.2	array	960
6.169.3.3	arrayOffset	960

6.169.3.4 asCharBuffer	961
6.169.3.5 asDoubleBuffer	961
6.169.3.6 asFloatBuffer	961
6.169.3.7 asIntBuffer	962
6.169.3.8 asLongBuffer	962
6.169.3.9 asReadOnlyBuffer	962
6.169.3.10asShortBuffer	963
6.169.3.11compact	963
6.169.3.12compareTo	963
6.169.3.13duplicate	963
6.169.3.14equals	964
6.169.3.15get	964
6.169.3.16get	964
6.169.3.17get	964
6.169.3.18get	965
6.169.3.19getChar	965
6.169.3.20getChar	966
6.169.3.21getDouble	966
6.169.3.22getDouble	966
6.169.3.23getFloat	967
6.169.3.24getFloat	967
6.169.3.25getInt	968
6.169.3.26getInt	968
6.169.3.27getLong	968
6.169.3.28getLong	969
6.169.3.29getShort	969
6.169.3.30getShort	969
6.169.3.31hasArray	970
6.169.3.32isReadOnly	970
6.169.3.33operator<	970
6.169.3.34operator==	970
6.169.3.35put	970
6.169.3.36put	971
6.169.3.37put	971
6.169.3.38put	972
6.169.3.39put	972

6.169.3.40	putChar	973
6.169.3.41	putChar	973
6.169.3.42	putDouble	973
6.169.3.43	putDouble	974
6.169.3.44	putFloat	974
6.169.3.45	putFloat	975
6.169.3.46	putInt	975
6.169.3.47	putInt	975
6.169.3.48	putLong	976
6.169.3.49	putLong	976
6.169.3.50	putShort	977
6.169.3.51	putShort	977
6.169.3.52	lice	977
6.169.3.53	oString	978
6.169.3.54	wrap	978
6.169.3.55	wrap	978
6.170	cms::BytesMessage Class Reference	979
6.170.1	Detailed Description	982
6.170.2	Constructor & Destructor Documentation	982
6.170.2.1	~BytesMessage	982
6.170.3	Member Function Documentation	982
6.170.3.1	clone	982
6.170.3.2	getBodyBytes	983
6.170.3.3	getBodyLength	983
6.170.3.4	readBoolean	983
6.170.3.5	readByte	984
6.170.3.6	readBytes	984
6.170.3.7	readBytes	984
6.170.3.8	readChar	985
6.170.3.9	readDouble	985
6.170.3.10	readFloat	986
6.170.3.11	readInt	986
6.170.3.12	readLong	986
6.170.3.13	readShort	987
6.170.3.14	readString	987
6.170.3.15	readUnsignedShort	987

6.170.3.16	readUTF	988
6.170.3.17	reset	988
6.170.3.18	setBodyBytes	988
6.170.3.19	writeBoolean	989
6.170.3.20	writeByte	989
6.170.3.21	writeBytes	989
6.170.3.22	writeBytes	990
6.170.3.23	writeChar	990
6.170.3.24	writeDouble	990
6.170.3.25	writeFloat	991
6.170.3.26	writeInt	991
6.170.3.27	writeLong	991
6.170.3.28	writeShort	992
6.170.3.29	writeString	992
6.170.3.30	writeUnsignedShort	992
6.170.3.31	writeUTF	993
6.171	activemq::cmsutil::CachedConsumer Class Reference	993
6.171.1	Detailed Description	994
6.171.2	Constructor & Destructor Documentation	994
6.171.2.1	CachedConsumer	994
6.171.2.2	CachedConsumer	994
6.171.2.3	~CachedConsumer	994
6.171.3	Member Function Documentation	994
6.171.3.1	close	994
6.171.3.2	getMessageListener	994
6.171.3.3	getMessageSelector	995
6.171.3.4	operator=	995
6.171.3.5	receive	995
6.171.3.6	receive	995
6.171.3.7	receiveNoWait	996
6.171.3.8	setMessageListener	996
6.172	activemq::cmsutil::CachedProducer Class Reference	996
6.172.1	Detailed Description	998
6.172.2	Constructor & Destructor Documentation	998
6.172.2.1	CachedProducer	998
6.172.2.2	CachedProducer	998

6.172.2.3 ~CachedProducer	998
6.172.3 Member Function Documentation	998
6.172.3.1 close	998
6.172.3.2 getDeliveryMode	998
6.172.3.3 getDisableMessageID	998
6.172.3.4 getDisableMessageTimeStamp	999
6.172.3.5 getPriority	999
6.172.3.6 getTimeToLive	999
6.172.3.7 operator=	1000
6.172.3.8 send	1000
6.172.3.9 send	1000
6.172.3.10 send	1001
6.172.3.11 send	1001
6.172.3.12 setDeliveryMode	1002
6.172.3.13 setDisableMessageID	1002
6.172.3.14 setDisableMessageTimeStamp	1002
6.172.3.15 setPriority	1002
6.172.3.16 setTimeToLive	1003
6.173 decaf::util::concurrent::Callable< V > Class Template Reference	1003
6.173.1 Detailed Description	1003
6.173.2 Constructor & Destructor Documentation	1004
6.173.2.1 ~Callable	1004
6.173.3 Member Function Documentation	1004
6.173.3.1 call	1004
6.174 decaf::util::concurrent::CancellationException Class Reference	1004
6.174.1 Constructor & Destructor Documentation	1005
6.174.1.1 CancellationException	1005
6.174.1.2 CancellationException	1005
6.174.1.3 CancellationException	1005
6.174.1.4 CancellationException	1005
6.174.1.5 CancellationException	1006
6.174.1.6 CancellationException	1006
6.174.1.7 ~CancellationException	1006
6.174.2 Member Function Documentation	1006
6.174.2.1 clone	1006
6.175 decaf::security::cert::Certificate Class Reference	1007

6.175.1 Detailed Description	1007
6.175.2 Constructor & Destructor Documentation	1008
6.175.2.1 ~Certificate	1008
6.175.3 Member Function Documentation	1008
6.175.3.1 equals	1008
6.175.3.2 getEncoded	1008
6.175.3.3 getPublicKey	1008
6.175.3.4 getPublicKey	1008
6.175.3.5 getType	1009
6.175.3.6 toString	1009
6.175.3.7 verify	1009
6.175.3.8 verify	1009
6.176decaf::security::cert::CertificateEncodingException Class Reference	1010
6.176.1 Constructor & Destructor Documentation	1010
6.176.1.1 CertificateEncodingException	1010
6.176.1.2 CertificateEncodingException	1011
6.176.1.3 CertificateEncodingException	1011
6.176.1.4 CertificateEncodingException	1011
6.176.1.5 ~CertificateEncodingException	1011
6.176.2 Member Function Documentation	1011
6.176.2.1 clone	1011
6.177decaf::security::cert::CertificateException Class Reference	1012
6.177.1 Constructor & Destructor Documentation	1012
6.177.1.1 CertificateException	1012
6.177.1.2 CertificateException	1012
6.177.1.3 CertificateException	1013
6.177.1.4 CertificateException	1013
6.177.1.5 ~CertificateException	1013
6.177.2 Member Function Documentation	1013
6.177.2.1 clone	1013
6.178decaf::security::cert::CertificateExpiredException Class Reference	1014
6.178.1 Constructor & Destructor Documentation	1014
6.178.1.1 CertificateExpiredException	1014
6.178.1.2 CertificateExpiredException	1014
6.178.1.3 CertificateExpiredException	1014
6.178.1.4 CertificateExpiredException	1015

6.178.1.5 ~CertificateExpiredException	1015
6.178.2 Member Function Documentation	1015
6.178.2.1 clone	1015
6.179decaf::security::cert::CertificateNotYetValidException Class Reference	1015
6.179.1 Constructor & Destructor Documentation	1016
6.179.1.1 CertificateNotYetValidException	1016
6.179.1.2 CertificateNotYetValidException	1016
6.179.1.3 CertificateNotYetValidException	1016
6.179.1.4 CertificateNotYetValidException	1017
6.179.1.5 ~CertificateNotYetValidException	1017
6.179.2 Member Function Documentation	1017
6.179.2.1 clone	1017
6.180decaf::security::cert::CertificateParsingException Class Reference	1017
6.180.1 Constructor & Destructor Documentation	1018
6.180.1.1 CertificateParsingException	1018
6.180.1.2 CertificateParsingException	1018
6.180.1.3 CertificateParsingException	1018
6.180.1.4 CertificateParsingException	1019
6.180.1.5 ~CertificateParsingException	1019
6.180.2 Member Function Documentation	1019
6.180.2.1 clone	1019
6.181decaf::lang::Character Class Reference	1019
6.181.1 Constructor & Destructor Documentation	1021
6.181.1.1 Character	1021
6.181.2 Member Function Documentation	1022
6.181.2.1 byteValue	1022
6.181.2.2 compareTo	1022
6.181.2.3 compareTo	1022
6.181.2.4 digit	1022
6.181.2.5 doubleValue	1023
6.181.2.6 equals	1023
6.181.2.7 equals	1023
6.181.2.8 floatValue	1023
6.181.2.9 intValue	1023
6.181.2.10sDigit	1024
6.181.2.11sISOControl	1024

6.181.2.12sLetter	1024
6.181.2.13sLetterOrDigit	1024
6.181.2.14sLowerCase	1024
6.181.2.15sUpperCase	1024
6.181.2.16sWhitespace	1024
6.181.2.17ongValue	1024
6.181.2.18operator<	1025
6.181.2.19operator<	1025
6.181.2.20operator==	1025
6.181.2.21operator==	1025
6.181.2.22shortValue	1026
6.181.2.23oString	1026
6.181.2.24valueOf	1026
6.181.3 Field Documentation	1026
6.181.3.1 MAX_RADIX	1026
6.181.3.2 MAX_VALUE	1026
6.181.3.3 MIN_RADIX	1026
6.181.3.4 MIN_VALUE	1026
6.181.3.5 SIZE	1027
6.182decaf::internal::nio::CharArrayBuffer Class Reference	1027
6.182.1 Constructor & Destructor Documentation	1031
6.182.1.1 CharArrayBuffer	1031
6.182.1.2 CharArrayBuffer	1031
6.182.1.3 CharArrayBuffer	1031
6.182.1.4 CharArrayBuffer	1032
6.182.1.5 ~CharArrayBuffer	1032
6.182.2 Member Function Documentation	1032
6.182.2.1 array	1032
6.182.2.2 arrayOffset	1033
6.182.2.3 asReadOnlyBuffer	1033
6.182.2.4 compact	1033
6.182.2.5 duplicate	1034
6.182.2.6 get	1034
6.182.2.7 get	1034
6.182.2.8 hasArray	1035
6.182.2.9 isReadOnly	1035

6.182.2.10put	1035
6.182.2.11put	1035
6.182.2.12setReadOnly	1036
6.182.2.13lice	1036
6.182.2.14subSequence	1036
6.182.3 Field Documentation	1037
6.182.3.1 _array	1037
6.182.3.2 length	1037
6.182.3.3 offset	1037
6.182.3.4 readOnly	1037
6.183decaf::nio::CharBuffer Class Reference	1037
6.183.1 Detailed Description	1040
6.183.2 Constructor & Destructor Documentation	1041
6.183.2.1 CharBuffer	1041
6.183.2.2 ~CharBuffer	1041
6.183.3 Member Function Documentation	1041
6.183.3.1 allocate	1041
6.183.3.2 append	1041
6.183.3.3 append	1042
6.183.3.4 append	1042
6.183.3.5 array	1043
6.183.3.6 arrayOffset	1043
6.183.3.7 asReadOnlyBuffer	1043
6.183.3.8 charAt	1044
6.183.3.9 compact	1044
6.183.3.10compareTo	1044
6.183.3.11duplicate	1044
6.183.3.12equals	1045
6.183.3.13get	1045
6.183.3.14get	1045
6.183.3.15get	1046
6.183.3.16get	1046
6.183.3.17hasArray	1047
6.183.3.18length	1047
6.183.3.19operator<	1047
6.183.3.20operator==	1047

6.183.3.21put	1047
6.183.3.22put	1048
6.183.3.23put	1048
6.183.3.24put	1049
6.183.3.25put	1049
6.183.3.26put	1050
6.183.3.27put	1050
6.183.3.28read	1050
6.183.3.29lice	1051
6.183.3.30subSequence	1051
6.183.3.31toString	1052
6.183.3.32wrap	1052
6.183.3.33wrap	1052
6.184decaf::lang::CharSequence Class Reference	1053
6.184.1 Detailed Description	1053
6.184.2 Constructor & Destructor Documentation	1054
6.184.2.1 ~CharSequence	1054
6.184.3 Member Function Documentation	1054
6.184.3.1 charAt	1054
6.184.3.2 length	1054
6.184.3.3 subSequence	1054
6.184.3.4 toString	1055
6.185decaf::util::zip::CheckedInputStream Class Reference	1055
6.185.1 Detailed Description	1056
6.185.2 Constructor & Destructor Documentation	1056
6.185.2.1 CheckedInputStream	1056
6.185.2.2 ~CheckedInputStream	1057
6.185.3 Member Function Documentation	1057
6.185.3.1 doReadArrayBounded	1057
6.185.3.2 doReadByte	1057
6.185.3.3 getChecksum	1057
6.185.3.4 skip	1057
6.186decaf::util::zip::CheckedOutputStream Class Reference	1058
6.186.1 Detailed Description	1058
6.186.2 Constructor & Destructor Documentation	1058
6.186.2.1 CheckedOutputStream	1058

6.186.2.2 ~CheckedOutputStream	1059
6.186.3 Member Function Documentation	1059
6.186.3.1 doWriteArrayBounded	1059
6.186.3.2 doWriteByte	1059
6.186.3.3 getChecksum	1059
6.187decaf::util::zip::Checksum Class Reference	1059
6.187.1 Detailed Description	1060
6.187.2 Constructor & Destructor Documentation	1060
6.187.2.1 ~Checksum	1060
6.187.3 Member Function Documentation	1060
6.187.3.1 getValue	1060
6.187.3.2 reset	1060
6.187.3.3 update	1061
6.187.3.4 update	1061
6.187.3.5 update	1061
6.187.3.6 update	1062
6.188decaf::lang::exceptions::ClassCastException Class Reference	1062
6.188.1 Constructor & Destructor Documentation	1063
6.188.1.1 ClassCastException	1063
6.188.1.2 ClassCastException	1063
6.188.1.3 ClassCastException	1063
6.188.1.4 ClassCastException	1063
6.188.1.5 ClassCastException	1063
6.188.1.6 ClassCastException	1064
6.188.1.7 ~ClassCastException	1064
6.188.2 Member Function Documentation	1064
6.188.2.1 clone	1064
6.189cms::Closeable Class Reference	1064
6.189.1 Detailed Description	1065
6.189.2 Constructor & Destructor Documentation	1065
6.189.2.1 ~Closeable	1065
6.189.3 Member Function Documentation	1065
6.189.3.1 close	1065
6.190decaf::io::Closeable Class Reference	1065
6.190.1 Detailed Description	1066
6.190.2 Constructor & Destructor Documentation	1066

6.190.2.1 ~Closeable	1066
6.190.3 Member Function Documentation	1066
6.190.3.1 close	1066
6.191activemq::transport::failover::CloseTransportsTask Class Reference	1066
6.191.1 Constructor & Destructor Documentation	1067
6.191.1.1 CloseTransportsTask	1067
6.191.1.2 ~CloseTransportsTask	1067
6.191.2 Member Function Documentation	1067
6.191.2.1 add	1067
6.191.2.2 isPending	1067
6.191.2.3 iterate	1067
6.192activemq::cmsutil::CmsAccessor Class Reference	1068
6.192.1 Detailed Description	1069
6.192.2 Constructor & Destructor Documentation	1069
6.192.2.1 CmsAccessor	1069
6.192.2.2 CmsAccessor	1069
6.192.2.3 ~CmsAccessor	1069
6.192.3 Member Function Documentation	1069
6.192.3.1 checkConnectionFactory	1069
6.192.3.2 createConnection	1069
6.192.3.3 createSession	1070
6.192.3.4 destroy	1070
6.192.3.5 getConnectionFactory	1070
6.192.3.6 getConnectionFactory	1070
6.192.3.7 getResourceLifecycleManager	1070
6.192.3.8 getResourceLifecycleManager	1070
6.192.3.9 getSessionAcknowledgeMode	1070
6.192.3.10init	1071
6.192.3.11operator=	1071
6.192.3.12setConnectionFactory	1071
6.192.3.13setSessionAcknowledgeMode	1071
6.193activemq::cmsutil::CmsDestinationAccessor Class Reference	1071
6.193.1 Detailed Description	1072
6.193.2 Constructor & Destructor Documentation	1073
6.193.2.1 CmsDestinationAccessor	1073
6.193.2.2 CmsDestinationAccessor	1073

6.193.2.3 ~CmsDestinationAccessor	1073
6.193.3 Member Function Documentation	1073
6.193.3.1 checkDestinationResolver	1073
6.193.3.2 destroy	1073
6.193.3.3 getDestinationResolver	1073
6.193.3.4 getDestinationResolver	1073
6.193.3.5 init	1073
6.193.3.6 isPubSubDomain	1074
6.193.3.7 operator=	1074
6.193.3.8 resolveDestinationName	1074
6.193.3.9 setDestinationResolver	1074
6.193.3.10 setPubSubDomain	1074
6.194cms::CMSException Class Reference	1074
6.194.1 Detailed Description	1075
6.194.2 Constructor & Destructor Documentation	1076
6.194.2.1 CMSException	1076
6.194.2.2 CMSException	1076
6.194.2.3 CMSException	1076
6.194.2.4 CMSException	1076
6.194.2.5 ~CMSException	1076
6.194.3 Member Function Documentation	1076
6.194.3.1 getCause	1076
6.194.3.2 getMessage	1076
6.194.3.3 getStackTrace	1076
6.194.3.4 getStackTraceString	1077
6.194.3.5 printStackTrace	1077
6.194.3.6 printStackTrace	1077
6.194.3.7 setMark	1077
6.194.3.8 what	1077
6.195activemq::util::CMSExceptionSupport Class Reference	1077
6.195.1 Constructor & Destructor Documentation	1078
6.195.1.1 ~CMSExceptionSupport	1078
6.195.2 Member Function Documentation	1078
6.195.2.1 create	1078
6.195.2.2 create	1078
6.195.2.3 createMessageEOFException	1078

6.195.2.4 createMessageFormatException	1078
6.196cms::CMSProperties Class Reference	1079
6.196.1 Detailed Description	1080
6.196.2 Constructor & Destructor Documentation	1080
6.196.2.1 ~CMSProperties	1080
6.196.3 Member Function Documentation	1080
6.196.3.1 clear	1080
6.196.3.2 clone	1080
6.196.3.3 copy	1080
6.196.3.4 getProperty	1080
6.196.3.5 getProperty	1081
6.196.3.6 hasProperty	1081
6.196.3.7 isEmpty	1081
6.196.3.8 remove	1081
6.196.3.9 setProperty	1082
6.196.3.10oArray	1082
6.196.3.11toString	1082
6.197cms::CMSSecurityException Class Reference	1082
6.197.1 Detailed Description	1083
6.197.2 Constructor & Destructor Documentation	1083
6.197.2.1 CMSSecurityException	1083
6.197.2.2 CMSSecurityException	1083
6.197.2.3 CMSSecurityException	1083
6.197.2.4 CMSSecurityException	1083
6.197.2.5 ~CMSSecurityException	1083
6.198activemq::cmsutil::CmsTemplate Class Reference	1083
6.198.1 Detailed Description	1087
6.198.2 Constructor & Destructor Documentation	1087
6.198.2.1 CmsTemplate	1087
6.198.2.2 CmsTemplate	1087
6.198.2.3 CmsTemplate	1087
6.198.2.4 ~CmsTemplate	1087
6.198.3 Member Function Documentation	1087
6.198.3.1 destroy	1087
6.198.3.2 execute	1087
6.198.3.3 execute	1088

6.198.3.4 execute	1088
6.198.3.5 execute	1088
6.198.3.6 getDefaultDestination	1089
6.198.3.7 getDefaultDestination	1089
6.198.3.8 getDefaultDestinationName	1089
6.198.3.9 getDeliveryMode	1089
6.198.3.10 getPriority	1089
6.198.3.11 getReceiveTimeout	1089
6.198.3.12 getTimeToLive	1089
6.198.3.13 init	1090
6.198.3.14 sExplicitQosEnabled	1090
6.198.3.15 sMessageIdEnabled	1090
6.198.3.16 sMessageTimestampEnabled	1090
6.198.3.17 sNoLocal	1090
6.198.3.18 operator=	1090
6.198.3.19 receive	1090
6.198.3.20 receive	1091
6.198.3.21 receive	1091
6.198.3.22 receiveSelected	1091
6.198.3.23 receiveSelected	1092
6.198.3.24 receiveSelected	1092
6.198.3.25 send	1092
6.198.3.26 send	1093
6.198.3.27 send	1093
6.198.3.28 setDefaultDestination	1093
6.198.3.29 setDefaultDestinationName	1093
6.198.3.30 setDeliveryMode	1094
6.198.3.31 setDeliveryPersistent	1094
6.198.3.32 setExplicitQosEnabled	1094
6.198.3.33 setMessageIdEnabled	1095
6.198.3.34 setMessageTimestampEnabled	1095
6.198.3.35 setNoLocal	1095
6.198.3.36 setPriority	1095
6.198.3.37 setPubSubDomain	1095
6.198.3.38 setReceiveTimeout	1095
6.198.3.39 setTimeToLive	1095

6.198.4 Friends And Related Function Documentation	1096
6.198.4.1 ProducerExecutor	1096
6.198.4.2 ReceiveExecutor	1096
6.198.4.3 ResolveProducerExecutor	1096
6.198.4.4 ResolveReceiveExecutor	1096
6.198.4.5 SendExecutor	1096
6.198.5 Field Documentation	1096
6.198.5.1 DEFAULT_PRIORITY	1096
6.198.5.2 DEFAULT_TIME_TO_LIVE	1096
6.198.5.3 RECEIVE_TIMEOUT_INDEFINITE_WAIT	1096
6.198.5.4 RECEIVE_TIMEOUT_NO_WAIT	1096
6.199 code Struct Reference	1096
6.199.1 Field Documentation	1097
6.199.1.1 bits	1097
6.199.1.2 op	1097
6.199.1.3 val	1097
6.200 decaf::util::Collection< E > Class Template Reference	1097
6.200.1 Detailed Description	1098
6.200.2 Constructor & Destructor Documentation	1099
6.200.2.1 ~Collection	1099
6.200.3 Member Function Documentation	1099
6.200.3.1 add	1099
6.200.3.2 addAll	1100
6.200.3.3 clear	1100
6.200.3.4 contains	1101
6.200.3.5 containsAll	1102
6.200.3.6 equals	1103
6.200.3.7 isEmpty	1103
6.200.3.8 remove	1104
6.200.3.9 removeAll	1105
6.200.3.10 retainAll	1105
6.200.3.11 size	1106
6.200.3.12 toArray	1107
6.201 activemq::commands::Command Class Reference	1107
6.201.1 Constructor & Destructor Documentation	1108
6.201.1.1 ~Command	1108

6.201.2 Member Function Documentation	1108
6.201.2.1 getCommandId	1108
6.201.2.2 isBrokerInfo	1109
6.201.2.3 isConnectionInfo	1109
6.201.2.4 isConsumerInfo	1109
6.201.2.5 isKeepAliveInfo	1109
6.201.2.6 isMessage	1109
6.201.2.7 isMessageAck	1109
6.201.2.8 isMessageDispatch	1109
6.201.2.9 isMessageDispatchNotification	1109
6.201.2.10sProducerAck	1110
6.201.2.11sProducerInfo	1110
6.201.2.12sRemoveInfo	1110
6.201.2.13sRemoveSubscriptionInfo	1110
6.201.2.14sResponse	1110
6.201.2.15sResponseRequired	1110
6.201.2.16sShutdownInfo	1110
6.201.2.17sTransactionInfo	1111
6.201.2.18sWireFormatInfo	1111
6.201.2.19setCommandId	1111
6.201.2.20setResponseRequired	1111
6.201.2.21toString	1111
6.201.2.22visit	1112
6.202activemq::state::CommandVisitor Class Reference	1113
6.202.1 Detailed Description	1114
6.202.2 Constructor & Destructor Documentation	1115
6.202.2.1 ~CommandVisitor	1115
6.202.3 Member Function Documentation	1115
6.202.3.1 processBeginTransaction	1115
6.202.3.2 processBrokerError	1115
6.202.3.3 processBrokerInfo	1115
6.202.3.4 processCommitTransactionOnePhase	1115
6.202.3.5 processCommitTransactionTwoPhase	1115
6.202.3.6 processConnectionControl	1116
6.202.3.7 processConnectionError	1116
6.202.3.8 processConnectionInfo	1116

6.202.3.9 processConsumerControl	1116
6.202.3.10 processConsumerInfo	1116
6.202.3.11 processControlCommand	1116
6.202.3.12 processDestinationInfo	1116
6.202.3.13 processEndTransaction	1116
6.202.3.14 processFlushCommand	1117
6.202.3.15 processForgetTransaction	1117
6.202.3.16 processKeepAliveInfo	1117
6.202.3.17 processMessage	1117
6.202.3.18 processMessageAck	1117
6.202.3.19 processMessageDispatch	1117
6.202.3.20 processMessageDispatchNotification	1117
6.202.3.21 processMessagePull	1117
6.202.3.22 processPrepareTransaction	1117
6.202.3.23 processProducerAck	1118
6.202.3.24 processProducerInfo	1118
6.202.3.25 processRecoverTransactions	1118
6.202.3.26 processRemoveConnection	1118
6.202.3.27 processRemoveConsumer	1118
6.202.3.28 processRemoveDestination	1118
6.202.3.29 processRemoveInfo	1118
6.202.3.30 processRemoveProducer	1118
6.202.3.31 processRemoveSession	1119
6.202.3.32 processRemoveSubscriptionInfo	1119
6.202.3.33 processReplayCommand	1119
6.202.3.34 processResponse	1119
6.202.3.35 processRollbackTransaction	1119
6.202.3.36 processSessionInfo	1119
6.202.3.37 processShutdownInfo	1119
6.202.3.38 processTransactionInfo	1119
6.202.3.39 processWireFormat	1120
6.203 activemq::state::CommandVisitorAdapter Class Reference	1120
6.203.1 Detailed Description	1122
6.203.2 Constructor & Destructor Documentation	1123
6.203.2.1 ~CommandVisitorAdapter	1123
6.203.3 Member Function Documentation	1123

6.203.3.1	processBeginTransaction	1123
6.203.3.2	processBrokerError	1123
6.203.3.3	processBrokerInfo	1123
6.203.3.4	processCommitTransactionOnePhase	1123
6.203.3.5	processCommitTransactionTwoPhase	1123
6.203.3.6	processConnectionControl	1123
6.203.3.7	processConnectionError	1123
6.203.3.8	processConnectionInfo	1123
6.203.3.9	processConsumerControl	1123
6.203.3.10	processConsumerInfo	1123
6.203.3.11	processControlCommand	1123
6.203.3.12	processDestinationInfo	1123
6.203.3.13	processEndTransaction	1123
6.203.3.14	processFlushCommand	1123
6.203.3.15	processForgetTransaction	1123
6.203.3.16	processKeepAliveInfo	1123
6.203.3.17	processMessage	1123
6.203.3.18	processMessageAck	1123
6.203.3.19	processMessageDispatch	1123
6.203.3.20	processMessageDispatchNotification	1123
6.203.3.21	processMessagePull	1123
6.203.3.22	processPrepareTransaction	1123
6.203.3.23	processProducerAck	1123
6.203.3.24	processProducerInfo	1123
6.203.3.25	processRecoverTransactions	1123
6.203.3.26	processRemoveConnection	1123
6.203.3.27	processRemoveConsumer	1123
6.203.3.28	processRemoveDestination	1123
6.203.3.29	processRemoveInfo	1123
6.203.3.30	processRemoveProducer	1124
6.203.3.31	processRemoveSession	1124
6.203.3.32	processRemoveSubscriptionInfo	1124
6.203.3.33	processReplayCommand	1124
6.203.3.34	processResponse	1124
6.203.3.35	processRollbackTransaction	1124
6.203.3.36	processSessionInfo	1124

6.203.3.37	processShutdownInfo	1124
6.203.3.38	processTransactionInfo	1124
6.203.3.39	processWireFormat	1125
6.204	decaf::lang::Comparable< T > Class Template Reference	1125
6.204.1	Detailed Description	1125
6.204.2	Constructor & Destructor Documentation	1126
6.204.2.1	~Comparable	1126
6.204.3	Member Function Documentation	1126
6.204.3.1	compareTo	1126
6.204.3.2	equals	1126
6.204.3.3	operator<	1127
6.204.3.4	operator==	1127
6.205	decaf::util::Comparator< T > Class Template Reference	1127
6.205.1	Detailed Description	1128
6.205.2	Constructor & Destructor Documentation	1128
6.205.2.1	~Comparator	1128
6.205.3	Member Function Documentation	1128
6.205.3.1	compare	1128
6.205.3.2	operator()	1129
6.206	activemq::util::CompositeData Class Reference	1129
6.206.1	Detailed Description	1130
6.206.2	Constructor & Destructor Documentation	1131
6.206.2.1	CompositeData	1131
6.206.2.2	~CompositeData	1131
6.206.3	Member Function Documentation	1131
6.206.3.1	getComponents	1131
6.206.3.2	getComponents	1131
6.206.3.3	getFragment	1131
6.206.3.4	getHost	1131
6.206.3.5	getParameters	1131
6.206.3.6	getPath	1131
6.206.3.7	getScheme	1131
6.206.3.8	setComponents	1131
6.206.3.9	setFragment	1131
6.206.3.10	setHost	1131
6.206.3.11	setParameters	1131

6.206.3.12	setPath	1131
6.206.3.13	setScheme	1131
6.206.3.14	oURI	1131
6.207	activemq::threads::CompositeTask Class Reference	1132
6.207.1	Detailed Description	1132
6.207.2	Constructor & Destructor Documentation	1132
6.207.2.1	~CompositeTask	1132
6.207.3	Member Function Documentation	1132
6.207.3.1	isPending	1132
6.208	activemq::threads::CompositeTaskRunner Class Reference	1133
6.208.1	Detailed Description	1133
6.208.2	Constructor & Destructor Documentation	1134
6.208.2.1	CompositeTaskRunner	1134
6.208.2.2	~CompositeTaskRunner	1134
6.208.3	Member Function Documentation	1134
6.208.3.1	addTask	1134
6.208.3.2	iterate	1134
6.208.3.3	removeTask	1134
6.208.3.4	run	1134
6.208.3.5	shutdown	1134
6.208.3.6	shutdown	1134
6.208.3.7	wakeup	1135
6.209	activemq::transport::CompositeTransport Class Reference	1135
6.209.1	Detailed Description	1135
6.209.2	Constructor & Destructor Documentation	1136
6.209.2.1	~CompositeTransport	1136
6.209.3	Member Function Documentation	1136
6.209.3.1	addURI	1136
6.209.3.2	removeURI	1136
6.210	decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference	1136
6.210.1	Detailed Description	1137
6.210.2	Constructor & Destructor Documentation	1137
6.210.2.1	~ConcurrentMap	1137
6.210.3	Member Function Documentation	1137
6.210.3.1	putIfAbsent	1137
6.210.3.2	remove	1138

6.210.3.3 replace	1139
6.210.3.4 replace	1140
6.211decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference	1140
6.211.1 Detailed Description	1144
6.211.2 Constructor & Destructor Documentation	1145
6.211.2.1 ConcurrentStlMap	1145
6.211.2.2 ConcurrentStlMap	1145
6.211.2.3 ConcurrentStlMap	1145
6.211.2.4 ~ConcurrentStlMap	1145
6.211.3 Member Function Documentation	1145
6.211.3.1 clear	1145
6.211.3.2 containsKey	1146
6.211.3.3 containsValue	1146
6.211.3.4 copy	1146
6.211.3.5 copy	1147
6.211.3.6 equals	1147
6.211.3.7 equals	1147
6.211.3.8 get	1147
6.211.3.9 get	1148
6.211.3.10isEmpty	1148
6.211.3.11keySet	1148
6.211.3.12lock	1149
6.211.3.13notify	1149
6.211.3.14notifyAll	1149
6.211.3.15put	1150
6.211.3.16putAll	1150
6.211.3.17putAll	1150
6.211.3.18putIfAbsent	1151
6.211.3.19remove	1151
6.211.3.20remove	1152
6.211.3.21replace	1152
6.211.3.22replace	1153
6.211.3.23size	1153
6.211.3.24tryLock	1154
6.211.3.25unlock	1154
6.211.3.26values	1154

6.211.3.27wait	1154
6.211.3.28wait	1155
6.211.3.29wait	1155
6.212decaf::util::concurrent::locks::Condition Class Reference	1156
6.212.1 Detailed Description	1157
6.212.2 Constructor & Destructor Documentation	1158
6.212.2.1 ~Condition	1158
6.212.3 Member Function Documentation	1158
6.212.3.1 await	1158
6.212.3.2 await	1159
6.212.3.3 awaitNanos	1160
6.212.3.4 awaitUninterruptibly	1161
6.212.3.5 awaitUntil	1162
6.212.3.6 signal	1162
6.212.3.7 signalAll	1162
6.213decaf::util::concurrent::ConditionHandle Class Reference	1162
6.213.1 Constructor & Destructor Documentation	1163
6.213.1.1 ConditionHandle	1163
6.213.1.2 ~ConditionHandle	1163
6.213.1.3 ConditionHandle	1163
6.213.1.4 ~ConditionHandle	1163
6.213.2 Field Documentation	1163
6.213.2.1 condition	1163
6.213.2.2 criticalSection	1163
6.213.2.3 generation	1163
6.213.2.4 mutex	1163
6.213.2.5 numWaiting	1163
6.213.2.6 numWake	1163
6.213.2.7 semaphore	1163
6.214decaf::internal::util::concurrent::ConditionImpl Class Reference	1163
6.214.1 Member Function Documentation	1164
6.214.1.1 create	1164
6.214.1.2 destroy	1164
6.214.1.3 notify	1165
6.214.1.4 notifyAll	1165
6.214.1.5 wait	1165

6.214.1.6 wait	1165
6.215decaf::net::ConnectException Class Reference	1165
6.215.1 Constructor & Destructor Documentation	1166
6.215.1.1 ConnectException	1166
6.215.1.2 ConnectException	1166
6.215.1.3 ConnectException	1166
6.215.1.4 ConnectException	1167
6.215.1.5 ConnectException	1167
6.215.1.6 ConnectException	1167
6.215.1.7 ~ConnectException	1167
6.215.2 Member Function Documentation	1167
6.215.2.1 clone	1167
6.216cms::Connection Class Reference	1168
6.216.1 Detailed Description	1169
6.216.2 Constructor & Destructor Documentation	1169
6.216.2.1 ~Connection	1169
6.216.3 Member Function Documentation	1169
6.216.3.1 close	1169
6.216.3.2 createSession	1170
6.216.3.3 createSession	1170
6.216.3.4 getClientID	1170
6.216.3.5 getExceptionListener	1170
6.216.3.6 getMetaData	1171
6.216.3.7 setClientID	1171
6.216.3.8 setExceptionListener	1171
6.217activemq::commands::ConnectionControl Class Reference	1172
6.217.1 Constructor & Destructor Documentation	1173
6.217.1.1 ConnectionControl	1173
6.217.1.2 ~ConnectionControl	1173
6.217.2 Member Function Documentation	1173
6.217.2.1 cloneDataStructure	1173
6.217.2.2 copyDataStructure	1173
6.217.2.3 equals	1174
6.217.2.4 getConnectedBrokers	1174
6.217.2.5 getConnectedBrokers	1174
6.217.2.6 getDataStructureType	1174

6.217.2.7	getReconnectTo	1175
6.217.2.8	getReconnectTo	1175
6.217.2.9	isClose	1175
6.217.2.10	isExit	1175
6.217.2.11	isFaultTolerant	1175
6.217.2.12	isRebalanceConnection	1175
6.217.2.13	isResume	1175
6.217.2.14	isSuspend	1175
6.217.2.15	setClose	1175
6.217.2.16	setConnectedBrokers	1175
6.217.2.17	setExit	1175
6.217.2.18	setFaultTolerant	1175
6.217.2.19	setRebalanceConnection	1175
6.217.2.20	setReconnectTo	1175
6.217.2.21	setResume	1175
6.217.2.22	setSuspend	1175
6.217.2.23	toString	1175
6.217.2.24	visit	1176
6.217.3	Field Documentation	1176
6.217.3.1	close	1176
6.217.3.2	connectedBrokers	1176
6.217.3.3	exit	1176
6.217.3.4	faultTolerant	1176
6.217.3.5	ID_CONNECTIONCONTROL	1176
6.217.3.6	rebalanceConnection	1176
6.217.3.7	reconnectTo	1176
6.217.3.8	resume	1176
6.217.3.9	suspend	1176
6.218	activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller Class Reference	1177
6.218.1	Detailed Description	1178
6.218.2	Constructor & Destructor Documentation	1178
6.218.2.1	ConnectionControlMarshaller	1178
6.218.2.2	~ConnectionControlMarshaller	1178
6.218.3	Member Function Documentation	1178
6.218.3.1	createObject	1178
6.218.3.2	getDataStructureType	1178

6.218.3.3 looseMarshal	1178
6.218.3.4 looseUnmarshal	1179
6.218.3.5 tightMarshal1	1179
6.218.3.6 tightMarshal2	1180
6.218.3.7 tightUnmarshal	1180
6.219activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller Class	
Reference	1180
6.219.1 Detailed Description	1181
6.219.2 Constructor & Destructor Documentation	1182
6.219.2.1 ConnectionControlMarshaller	1182
6.219.2.2 ~ConnectionControlMarshaller	1182
6.219.3 Member Function Documentation	1182
6.219.3.1 createObject	1182
6.219.3.2 getDataStructureType	1182
6.219.3.3 looseMarshal	1182
6.219.3.4 looseUnmarshal	1183
6.219.3.5 tightMarshal1	1183
6.219.3.6 tightMarshal2	1183
6.219.3.7 tightUnmarshal	1184
6.220activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller Class	
Reference	1184
6.220.1 Detailed Description	1185
6.220.2 Constructor & Destructor Documentation	1186
6.220.2.1 ConnectionControlMarshaller	1186
6.220.2.2 ~ConnectionControlMarshaller	1186
6.220.3 Member Function Documentation	1186
6.220.3.1 createObject	1186
6.220.3.2 getDataStructureType	1186
6.220.3.3 looseMarshal	1186
6.220.3.4 looseUnmarshal	1187
6.220.3.5 tightMarshal1	1187
6.220.3.6 tightMarshal2	1187
6.220.3.7 tightUnmarshal	1188
6.221activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller Class	
Reference	1188
6.221.1 Detailed Description	1189
6.221.2 Constructor & Destructor Documentation	1190

6.221.2.1	ConnectionControlMarshaller	1190
6.221.2.2	~ConnectionControlMarshaller	1190
6.221.3	Member Function Documentation	1190
6.221.3.1	createObject	1190
6.221.3.2	getDataStructureType	1190
6.221.3.3	looseMarshal	1190
6.221.3.4	looseUnmarshal	1191
6.221.3.5	tightMarshal1	1191
6.221.3.6	tightMarshal2	1191
6.221.3.7	tightUnmarshal	1192
6.222	activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller Class	
	Reference	1192
6.222.1	Detailed Description	1193
6.222.2	Constructor & Destructor Documentation	1194
6.222.2.1	ConnectionControlMarshaller	1194
6.222.2.2	~ConnectionControlMarshaller	1194
6.222.3	Member Function Documentation	1194
6.222.3.1	createObject	1194
6.222.3.2	getDataStructureType	1194
6.222.3.3	looseMarshal	1194
6.222.3.4	looseUnmarshal	1195
6.222.3.5	tightMarshal1	1195
6.222.3.6	tightMarshal2	1195
6.222.3.7	tightUnmarshal	1196
6.223	activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller Class	
	Reference	1196
6.223.1	Detailed Description	1197
6.223.2	Constructor & Destructor Documentation	1198
6.223.2.1	ConnectionControlMarshaller	1198
6.223.2.2	~ConnectionControlMarshaller	1198
6.223.3	Member Function Documentation	1198
6.223.3.1	createObject	1198
6.223.3.2	getDataStructureType	1198
6.223.3.3	looseMarshal	1198
6.223.3.4	looseUnmarshal	1199
6.223.3.5	tightMarshal1	1199
6.223.3.6	tightMarshal2	1199

6.223.3.7 tightUnmarshal	1200
6.224activemq::commands::ConnectionError Class Reference	1200
6.224.1 Constructor & Destructor Documentation	1202
6.224.1.1 ConnectionError	1202
6.224.1.2 ~ConnectionError	1202
6.224.2 Member Function Documentation	1202
6.224.2.1 cloneDataStructure	1202
6.224.2.2 copyDataStructure	1202
6.224.2.3 equals	1202
6.224.2.4 getConnectionId	1203
6.224.2.5 getConnectionId	1203
6.224.2.6 getDataStructureType	1203
6.224.2.7 getException	1203
6.224.2.8 getException	1203
6.224.2.9 setConnectionId	1203
6.224.2.10setException	1203
6.224.2.11toString	1203
6.224.2.12visit	1203
6.224.3 Field Documentation	1204
6.224.3.1 connectionId	1204
6.224.3.2 exception	1204
6.224.3.3 ID_CONNECTIONERROR	1204
6.225activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller Class Reference	1204
6.225.1 Detailed Description	1205
6.225.2 Constructor & Destructor Documentation	1205
6.225.2.1 ConnectionErrorMarshaller	1205
6.225.2.2 ~ConnectionErrorMarshaller	1205
6.225.3 Member Function Documentation	1205
6.225.3.1 createObject	1205
6.225.3.2 getDataStructureType	1206
6.225.3.3 looseMarshal	1206
6.225.3.4 looseUnmarshal	1206
6.225.3.5 tightMarshal1	1207
6.225.3.6 tightMarshal2	1207
6.225.3.7 tightUnmarshal	1207

6.226	activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	Class	
	Reference		1208
6.226.1	Detailed Description		1209
6.226.2	Constructor & Destructor Documentation		1209
6.226.2.1	ConnectionErrorMarshaller		1209
6.226.2.2	~ConnectionErrorMarshaller		1209
6.226.3	Member Function Documentation		1209
6.226.3.1	createObject		1209
6.226.3.2	getDataStructureType		1209
6.226.3.3	looseMarshal		1210
6.226.3.4	looseUnmarshal		1210
6.226.3.5	tightMarshal1		1210
6.226.3.6	tightMarshal2		1211
6.226.3.7	tightUnmarshal		1211
6.227	activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	Class	
	Reference		1212
6.227.1	Detailed Description		1213
6.227.2	Constructor & Destructor Documentation		1213
6.227.2.1	ConnectionErrorMarshaller		1213
6.227.2.2	~ConnectionErrorMarshaller		1213
6.227.3	Member Function Documentation		1213
6.227.3.1	createObject		1213
6.227.3.2	getDataStructureType		1213
6.227.3.3	looseMarshal		1214
6.227.3.4	looseUnmarshal		1214
6.227.3.5	tightMarshal1		1214
6.227.3.6	tightMarshal2		1215
6.227.3.7	tightUnmarshal		1215
6.228	activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	Class	
	Reference		1216
6.228.1	Detailed Description		1217
6.228.2	Constructor & Destructor Documentation		1217
6.228.2.1	ConnectionErrorMarshaller		1217
6.228.2.2	~ConnectionErrorMarshaller		1217
6.228.3	Member Function Documentation		1217
6.228.3.1	createObject		1217
6.228.3.2	getDataStructureType		1217

6.228.3.3 looseMarshal	1218
6.228.3.4 looseUnmarshal	1218
6.228.3.5 tightMarshal1	1218
6.228.3.6 tightMarshal2	1219
6.228.3.7 tightUnmarshal	1219
6.229activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller Class	
Reference	1220
6.229.1 Detailed Description	1221
6.229.2 Constructor & Destructor Documentation	1221
6.229.2.1 ConnectionErrorMarshaller	1221
6.229.2.2 ~ConnectionErrorMarshaller	1221
6.229.3 Member Function Documentation	1221
6.229.3.1 createObject	1221
6.229.3.2 getDataStructureType	1221
6.229.3.3 looseMarshal	1222
6.229.3.4 looseUnmarshal	1222
6.229.3.5 tightMarshal1	1222
6.229.3.6 tightMarshal2	1223
6.229.3.7 tightUnmarshal	1223
6.230activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller Class	
Reference	1224
6.230.1 Detailed Description	1225
6.230.2 Constructor & Destructor Documentation	1225
6.230.2.1 ConnectionErrorMarshaller	1225
6.230.2.2 ~ConnectionErrorMarshaller	1225
6.230.3 Member Function Documentation	1225
6.230.3.1 createObject	1225
6.230.3.2 getDataStructureType	1225
6.230.3.3 looseMarshal	1226
6.230.3.4 looseUnmarshal	1226
6.230.3.5 tightMarshal1	1226
6.230.3.6 tightMarshal2	1227
6.230.3.7 tightUnmarshal	1227
6.231cms::ConnectionFactory Class Reference	1228
6.231.1 Detailed Description	1228
6.231.2 Constructor & Destructor Documentation	1229
6.231.2.1 ~ConnectionFactory	1229

6.231.3 Member Function Documentation	1229
6.231.3.1 createCMSConnectionFactory	1229
6.231.3.2 createConnection	1229
6.231.3.3 createConnection	1230
6.231.3.4 createConnection	1230
6.232activemq::commands::ConnectionId Class Reference	1231
6.232.1 Member Typedef Documentation	1232
6.232.1.1 COMPARATOR	1232
6.232.2 Constructor & Destructor Documentation	1232
6.232.2.1 ConnectionId	1232
6.232.2.2 ConnectionId	1232
6.232.2.3 ConnectionId	1232
6.232.2.4 ConnectionId	1232
6.232.2.5 ConnectionId	1232
6.232.2.6 ~ConnectionId	1232
6.232.3 Member Function Documentation	1232
6.232.3.1 cloneDataStructure	1232
6.232.3.2 compareTo	1232
6.232.3.3 copyDataStructure	1232
6.232.3.4 equals	1233
6.232.3.5 equals	1233
6.232.3.6 getDataStructureType	1233
6.232.3.7 getValue	1233
6.232.3.8 getValue	1233
6.232.3.9 operator<	1233
6.232.3.10operator=	1233
6.232.3.11operator==	1233
6.232.3.12setValue	1233
6.232.3.13toString	1233
6.232.4 Field Documentation	1234
6.232.4.1 ID_CONNECTIONID	1234
6.232.4.2 value	1234
6.233activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller Class Refer- ence	1234
6.233.1 Detailed Description	1235
6.233.2 Constructor & Destructor Documentation	1235
6.233.2.1 ConnectionIdMarshaller	1235

6.233.2.2 <code>~ConnectionIdMarshaller</code>	1235
6.233.3 Member Function Documentation	1235
6.233.3.1 <code>createObject</code>	1235
6.233.3.2 <code>getDataStructureType</code>	1235
6.233.3.3 <code>looseMarshal</code>	1236
6.233.3.4 <code>looseUnmarshal</code>	1236
6.233.3.5 <code>tightMarshal1</code>	1236
6.233.3.6 <code>tightMarshal2</code>	1237
6.233.3.7 <code>tightUnmarshal</code>	1237
6.234 <code>activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller</code> Class Reference	1238
6.234.1 Detailed Description	1239
6.234.2 Constructor & Destructor Documentation	1239
6.234.2.1 <code>ConnectionIdMarshaller</code>	1239
6.234.2.2 <code>~ConnectionIdMarshaller</code>	1239
6.234.3 Member Function Documentation	1239
6.234.3.1 <code>createObject</code>	1239
6.234.3.2 <code>getDataStructureType</code>	1239
6.234.3.3 <code>looseMarshal</code>	1239
6.234.3.4 <code>looseUnmarshal</code>	1240
6.234.3.5 <code>tightMarshal1</code>	1240
6.234.3.6 <code>tightMarshal2</code>	1241
6.234.3.7 <code>tightUnmarshal</code>	1241
6.235 <code>activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller</code> Class Reference	1241
6.235.1 Detailed Description	1242
6.235.2 Constructor & Destructor Documentation	1243
6.235.2.1 <code>ConnectionIdMarshaller</code>	1243
6.235.2.2 <code>~ConnectionIdMarshaller</code>	1243
6.235.3 Member Function Documentation	1243
6.235.3.1 <code>createObject</code>	1243
6.235.3.2 <code>getDataStructureType</code>	1243
6.235.3.3 <code>looseMarshal</code>	1243
6.235.3.4 <code>looseUnmarshal</code>	1244
6.235.3.5 <code>tightMarshal1</code>	1244
6.235.3.6 <code>tightMarshal2</code>	1244
6.235.3.7 <code>tightUnmarshal</code>	1245

6.236activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller Class Reference	1245
6.236.1 Detailed Description	1246
6.236.2 Constructor & Destructor Documentation	1247
6.236.2.1 ConnectionIdMarshaller	1247
6.236.2.2 ~ConnectionIdMarshaller	1247
6.236.3 Member Function Documentation	1247
6.236.3.1 createObject	1247
6.236.3.2 getDataStructureType	1247
6.236.3.3 looseMarshal	1247
6.236.3.4 looseUnmarshal	1248
6.236.3.5 tightMarshal1	1248
6.236.3.6 tightMarshal2	1248
6.236.3.7 tightUnmarshal	1249
6.237activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller Class Reference	1249
6.237.1 Detailed Description	1250
6.237.2 Constructor & Destructor Documentation	1251
6.237.2.1 ConnectionIdMarshaller	1251
6.237.2.2 ~ConnectionIdMarshaller	1251
6.237.3 Member Function Documentation	1251
6.237.3.1 createObject	1251
6.237.3.2 getDataStructureType	1251
6.237.3.3 looseMarshal	1251
6.237.3.4 looseUnmarshal	1252
6.237.3.5 tightMarshal1	1252
6.237.3.6 tightMarshal2	1252
6.237.3.7 tightUnmarshal	1253
6.238activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller Class Reference	1253
6.238.1 Detailed Description	1254
6.238.2 Constructor & Destructor Documentation	1255
6.238.2.1 ConnectionIdMarshaller	1255
6.238.2.2 ~ConnectionIdMarshaller	1255
6.238.3 Member Function Documentation	1255
6.238.3.1 createObject	1255
6.238.3.2 getDataStructureType	1255

6.238.3.3 looseMarshal	1255
6.238.3.4 looseUnmarshal	1256
6.238.3.5 tightMarshal1	1256
6.238.3.6 tightMarshal2	1256
6.238.3.7 tightUnmarshal	1257
6.239activemq::commands::ConnectionInfo Class Reference	1257
6.239.1 Constructor & Destructor Documentation	1259
6.239.1.1 ConnectionInfo	1259
6.239.1.2 ~ConnectionInfo	1259
6.239.2 Member Function Documentation	1259
6.239.2.1 cloneDataStructure	1259
6.239.2.2 copyDataStructure	1259
6.239.2.3 createRemoveCommand	1260
6.239.2.4 equals	1260
6.239.2.5 getBrokerPath	1260
6.239.2.6 getBrokerPath	1260
6.239.2.7 getClientId	1260
6.239.2.8 getClientId	1260
6.239.2.9 getConnectionId	1260
6.239.2.10getConnectionId	1260
6.239.2.11getDataStructureType	1260
6.239.2.12getPassword	1261
6.239.2.13getPassword	1261
6.239.2.14getUserName	1261
6.239.2.15getUserName	1261
6.239.2.16sBrokerMasterConnector	1261
6.239.2.17sClientMaster	1261
6.239.2.18sConnectionInfo	1261
6.239.2.19sFaultTolerant	1262
6.239.2.20sManageable	1262
6.239.2.21setBrokerMasterConnector	1262
6.239.2.22setBrokerPath	1262
6.239.2.23setClientId	1262
6.239.2.24setClientMaster	1262
6.239.2.25setConnectionId	1262
6.239.2.26setFaultTolerant	1262

6.239.2.27	setManageable	1262
6.239.2.28	setPassword	1262
6.239.2.29	setUserName	1262
6.239.2.30	toString	1262
6.239.2.31	visit	1262
6.239.3	Field Documentation	1263
6.239.3.1	brokerMasterConnector	1263
6.239.3.2	brokerPath	1263
6.239.3.3	clientId	1263
6.239.3.4	clientMaster	1263
6.239.3.5	connectionId	1263
6.239.3.6	faultTolerant	1263
6.239.3.7	ID_CONNECTIONINFO	1263
6.239.3.8	manageable	1263
6.239.3.9	password	1263
6.239.3.10	userName	1263
6.240	activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller Class Reference	1263
6.240.1	Detailed Description	1264
6.240.2	Constructor & Destructor Documentation	1265
6.240.2.1	ConnectionInfoMarshaller	1265
6.240.2.2	~ConnectionInfoMarshaller	1265
6.240.3	Member Function Documentation	1265
6.240.3.1	createObject	1265
6.240.3.2	getDataStructureType	1265
6.240.3.3	looseMarshal	1265
6.240.3.4	looseUnmarshal	1266
6.240.3.5	tightMarshal1	1266
6.240.3.6	tightMarshal2	1266
6.240.3.7	tightUnmarshal	1267
6.241	activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller Class Reference	1267
6.241.1	Detailed Description	1268
6.241.2	Constructor & Destructor Documentation	1269
6.241.2.1	ConnectionInfoMarshaller	1269
6.241.2.2	~ConnectionInfoMarshaller	1269
6.241.3	Member Function Documentation	1269

6.241.3.1 createObject	1269
6.241.3.2 getDataStructureType	1269
6.241.3.3 looseMarshal	1269
6.241.3.4 looseUnmarshal	1270
6.241.3.5 tightMarshal1	1270
6.241.3.6 tightMarshal2	1270
6.241.3.7 tightUnmarshal	1271
6.242activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller Class Reference	1271
6.242.1 Detailed Description	1272
6.242.2 Constructor & Destructor Documentation	1273
6.242.2.1 ConnectionInfoMarshaller	1273
6.242.2.2 ~ConnectionInfoMarshaller	1273
6.242.3 Member Function Documentation	1273
6.242.3.1 createObject	1273
6.242.3.2 getDataStructureType	1273
6.242.3.3 looseMarshal	1273
6.242.3.4 looseUnmarshal	1274
6.242.3.5 tightMarshal1	1274
6.242.3.6 tightMarshal2	1274
6.242.3.7 tightUnmarshal	1275
6.243activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller Class Reference	1275
6.243.1 Detailed Description	1276
6.243.2 Constructor & Destructor Documentation	1277
6.243.2.1 ConnectionInfoMarshaller	1277
6.243.2.2 ~ConnectionInfoMarshaller	1277
6.243.3 Member Function Documentation	1277
6.243.3.1 createObject	1277
6.243.3.2 getDataStructureType	1277
6.243.3.3 looseMarshal	1277
6.243.3.4 looseUnmarshal	1278
6.243.3.5 tightMarshal1	1278
6.243.3.6 tightMarshal2	1278
6.243.3.7 tightUnmarshal	1279
6.244activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller Class Reference	1279

6.244.1 Detailed Description	1280
6.244.2 Constructor & Destructor Documentation	1281
6.244.2.1 ConnectionInfoMarshaller	1281
6.244.2.2 ~ConnectionInfoMarshaller	1281
6.244.3 Member Function Documentation	1281
6.244.3.1 createObject	1281
6.244.3.2 getDataStructureType	1281
6.244.3.3 looseMarshal	1281
6.244.3.4 looseUnmarshal	1282
6.244.3.5 tightMarshal1	1282
6.244.3.6 tightMarshal2	1282
6.244.3.7 tightUnmarshal	1283
6.245activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller Class Reference	1283
6.245.1 Detailed Description	1284
6.245.2 Constructor & Destructor Documentation	1285
6.245.2.1 ConnectionInfoMarshaller	1285
6.245.2.2 ~ConnectionInfoMarshaller	1285
6.245.3 Member Function Documentation	1285
6.245.3.1 createObject	1285
6.245.3.2 getDataStructureType	1285
6.245.3.3 looseMarshal	1285
6.245.3.4 looseUnmarshal	1286
6.245.3.5 tightMarshal1	1286
6.245.3.6 tightMarshal2	1286
6.245.3.7 tightUnmarshal	1287
6.246cms::ConnectionMetaData Class Reference	1287
6.246.1 Detailed Description	1288
6.246.2 Constructor & Destructor Documentation	1288
6.246.2.1 ~ConnectionMetaData	1288
6.246.3 Member Function Documentation	1288
6.246.3.1 getCMSMajorVersion	1288
6.246.3.2 getCMSMinorVersion	1289
6.246.3.3 getCMSProviderName	1289
6.246.3.4 getCMSVersion	1289
6.246.3.5 getCMSXPropertyNames	1290
6.246.3.6 getProviderMajorVersion	1290

6.246.3.7	getProviderMinorVersion	1290
6.246.3.8	getProviderVersion	1290
6.247	activemq::state::ConnectionState Class Reference	1291
6.247.1	Constructor & Destructor Documentation	1292
6.247.1.1	ConnectionState	1292
6.247.1.2	~ConnectionState	1292
6.247.2	Member Function Documentation	1292
6.247.2.1	addSession	1292
6.247.2.2	addTempDestination	1292
6.247.2.3	addTransactionState	1292
6.247.2.4	checkShutdown	1292
6.247.2.5	getInfo	1292
6.247.2.6	getRecoveringPullConsumers	1292
6.247.2.7	getSessionState	1292
6.247.2.8	getSessionStates	1292
6.247.2.9	getTempDesinations	1292
6.247.2.10	getTransactionState	1292
6.247.2.11	getTransactionStates	1293
6.247.2.12	sConnectionInterruptProcessingComplete	1293
6.247.2.13	removeSession	1293
6.247.2.14	removeTempDestination	1293
6.247.2.15	removeTransactionState	1293
6.247.2.16	reset	1293
6.247.2.17	setConnectionInterruptProcessingComplete	1293
6.247.2.18	shutdown	1293
6.247.2.19	oString	1293
6.248	activemq::state::ConnectionStateTracker Class Reference	1293
6.248.1	Constructor & Destructor Documentation	1296
6.248.1.1	ConnectionStateTracker	1296
6.248.1.2	~ConnectionStateTracker	1296
6.248.2	Member Function Documentation	1296
6.248.2.1	connectionInterruptProcessingComplete	1296
6.248.2.2	getMaxCacheSize	1296
6.248.2.3	isRestoreConsumers	1296
6.248.2.4	isRestoreProducers	1296
6.248.2.5	isRestoreSessions	1296

6.248.2.6	isRestoreTransaction	1296
6.248.2.7	isTrackMessages	1296
6.248.2.8	isTrackTransactionProducers	1296
6.248.2.9	isTrackTransactions	1296
6.248.2.10	processBeginTransaction	1296
6.248.2.11	processCommitTransactionOnePhase	1296
6.248.2.12	processCommitTransactionTwoPhase	1297
6.248.2.13	processConnectionInfo	1297
6.248.2.14	processConsumerInfo	1297
6.248.2.15	processDestinationInfo	1297
6.248.2.16	processEndTransaction	1297
6.248.2.17	processMessage	1297
6.248.2.18	processMessageAck	1297
6.248.2.19	processPrepareTransaction	1298
6.248.2.20	processProducerInfo	1298
6.248.2.21	processRemoveConnection	1298
6.248.2.22	processRemoveConsumer	1298
6.248.2.23	processRemoveDestination	1298
6.248.2.24	processRemoveProducer	1298
6.248.2.25	processRemoveSession	1298
6.248.2.26	processRollbackTransaction	1299
6.248.2.27	processSessionInfo	1299
6.248.2.28	restore	1300
6.248.2.29	setMaxCacheSize	1300
6.248.2.30	setRestoreConsumers	1300
6.248.2.31	setRestoreProducers	1300
6.248.2.32	setRestoreSessions	1300
6.248.2.33	setRestoreTransaction	1300
6.248.2.34	setTrackMessages	1300
6.248.2.35	setTrackTransactionProducers	1300
6.248.2.36	setTrackTransactions	1300
6.248.2.37	track	1300
6.248.2.38	rackBack	1300
6.248.2.39	ransportInterrupted	1300
6.248.3	Friends And Related Function Documentation	1300
6.248.3.1	RemoveTransactionAction	1300

6.249	decaf::util::logging::ConsoleHandler Class Reference	1300
6.249.1	Detailed Description	1301
6.249.2	Constructor & Destructor Documentation	1301
6.249.2.1	ConsoleHandler	1301
6.249.2.2	~ConsoleHandler	1301
6.249.3	Member Function Documentation	1301
6.249.3.1	close	1301
6.249.3.2	publish	1302
6.250	activemq::commands::ConsumerControl Class Reference	1302
6.250.1	Constructor & Destructor Documentation	1303
6.250.1.1	ConsumerControl	1303
6.250.1.2	~ConsumerControl	1303
6.250.2	Member Function Documentation	1303
6.250.2.1	cloneDataStructure	1303
6.250.2.2	copyDataStructure	1304
6.250.2.3	equals	1304
6.250.2.4	getConsumerId	1304
6.250.2.5	getConsumerId	1304
6.250.2.6	getDataStructureType	1304
6.250.2.7	getDestination	1305
6.250.2.8	getDestination	1305
6.250.2.9	getPrefetch	1305
6.250.2.10	isClose	1305
6.250.2.11	isFlush	1305
6.250.2.12	isStart	1305
6.250.2.13	isStop	1305
6.250.2.14	setClose	1305
6.250.2.15	setConsumerId	1305
6.250.2.16	setDestination	1305
6.250.2.17	setFlush	1305
6.250.2.18	setPrefetch	1305
6.250.2.19	setStart	1305
6.250.2.20	setStop	1305
6.250.2.21	toString	1305
6.250.2.22	visit	1306
6.250.3	Field Documentation	1306

6.250.3.1 close	1306
6.250.3.2 consumerId	1306
6.250.3.3 destination	1306
6.250.3.4 flush	1306
6.250.3.5 ID_CONSUMERCONTROL	1306
6.250.3.6 prefetch	1306
6.250.3.7 start	1306
6.250.3.8 stop	1306
6.251activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller Class	
Reference	1306
6.251.1 Detailed Description	1307
6.251.2 Constructor & Destructor Documentation	1308
6.251.2.1 ConsumerControlMarshaller	1308
6.251.2.2 ~ConsumerControlMarshaller	1308
6.251.3 Member Function Documentation	1308
6.251.3.1 createObject	1308
6.251.3.2 getDataStructureType	1308
6.251.3.3 looseMarshal	1308
6.251.3.4 looseUnmarshal	1309
6.251.3.5 tightMarshal1	1309
6.251.3.6 tightMarshal2	1309
6.251.3.7 tightUnmarshal	1310
6.252activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller Class	
Reference	1310
6.252.1 Detailed Description	1311
6.252.2 Constructor & Destructor Documentation	1312
6.252.2.1 ConsumerControlMarshaller	1312
6.252.2.2 ~ConsumerControlMarshaller	1312
6.252.3 Member Function Documentation	1312
6.252.3.1 createObject	1312
6.252.3.2 getDataStructureType	1312
6.252.3.3 looseMarshal	1312
6.252.3.4 looseUnmarshal	1313
6.252.3.5 tightMarshal1	1313
6.252.3.6 tightMarshal2	1313
6.252.3.7 tightUnmarshal	1314

6.253	activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	Class	
	Reference		1314
6.253.1	Detailed Description		1315
6.253.2	Constructor & Destructor Documentation		1316
6.253.2.1	ConsumerControlMarshaller		1316
6.253.2.2	~ConsumerControlMarshaller		1316
6.253.3	Member Function Documentation		1316
6.253.3.1	createObject		1316
6.253.3.2	getDataStructureType		1316
6.253.3.3	looseMarshal		1316
6.253.3.4	looseUnmarshal		1317
6.253.3.5	tightMarshal1		1317
6.253.3.6	tightMarshal2		1317
6.253.3.7	tightUnmarshal		1318
6.254	activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	Class	
	Reference		1318
6.254.1	Detailed Description		1319
6.254.2	Constructor & Destructor Documentation		1320
6.254.2.1	ConsumerControlMarshaller		1320
6.254.2.2	~ConsumerControlMarshaller		1320
6.254.3	Member Function Documentation		1320
6.254.3.1	createObject		1320
6.254.3.2	getDataStructureType		1320
6.254.3.3	looseMarshal		1320
6.254.3.4	looseUnmarshal		1321
6.254.3.5	tightMarshal1		1321
6.254.3.6	tightMarshal2		1321
6.254.3.7	tightUnmarshal		1322
6.255	activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller	Class	
	Reference		1322
6.255.1	Detailed Description		1323
6.255.2	Constructor & Destructor Documentation		1324
6.255.2.1	ConsumerControlMarshaller		1324
6.255.2.2	~ConsumerControlMarshaller		1324
6.255.3	Member Function Documentation		1324
6.255.3.1	createObject		1324
6.255.3.2	getDataStructureType		1324

6.255.3.3 looseMarshal	1324
6.255.3.4 looseUnmarshal	1325
6.255.3.5 tightMarshal1	1325
6.255.3.6 tightMarshal2	1325
6.255.3.7 tightUnmarshal	1326
6.256activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller Class Reference	1326
6.256.1 Detailed Description	1327
6.256.2 Constructor & Destructor Documentation	1328
6.256.2.1 ConsumerControlMarshaller	1328
6.256.2.2 ~ConsumerControlMarshaller	1328
6.256.3 Member Function Documentation	1328
6.256.3.1 createObject	1328
6.256.3.2 getDataStructureType	1328
6.256.3.3 looseMarshal	1328
6.256.3.4 looseUnmarshal	1329
6.256.3.5 tightMarshal1	1329
6.256.3.6 tightMarshal2	1329
6.256.3.7 tightUnmarshal	1330
6.257activemq::commands::ConsumerId Class Reference	1330
6.257.1 Member Typedef Documentation	1332
6.257.1.1 COMPARATOR	1332
6.257.2 Constructor & Destructor Documentation	1332
6.257.2.1 ConsumerId	1332
6.257.2.2 ConsumerId	1332
6.257.2.3 ConsumerId	1332
6.257.2.4 ~ConsumerId	1332
6.257.3 Member Function Documentation	1332
6.257.3.1 cloneDataStructure	1332
6.257.3.2 compareTo	1332
6.257.3.3 copyDataStructure	1332
6.257.3.4 equals	1332
6.257.3.5 equals	1333
6.257.3.6 getConnectionId	1333
6.257.3.7 getConnectionId	1333
6.257.3.8 getDataStructureType	1333
6.257.3.9 getParentId	1333

6.257.3.10	getSessionId	1333
6.257.3.11	getValue	1333
6.257.3.12	operator<	1333
6.257.3.13	operator=	1333
6.257.3.14	operator==	1333
6.257.3.15	setConnectionId	1333
6.257.3.16	setSessionId	1333
6.257.3.17	setValue	1333
6.257.3.18	toString	1333
6.257.4	Field Documentation	1334
6.257.4.1	connectionId	1334
6.257.4.2	ID_CONSUMERID	1334
6.257.4.3	sessionId	1334
6.257.4.4	value	1334
6.258	activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller Class Reference	1334
6.258.1	Detailed Description	1335
6.258.2	Constructor & Destructor Documentation	1335
6.258.2.1	ConsumerIdMarshaller	1335
6.258.2.2	~ConsumerIdMarshaller	1335
6.258.3	Member Function Documentation	1335
6.258.3.1	createObject	1335
6.258.3.2	getDataStructureType	1335
6.258.3.3	looseMarshal	1336
6.258.3.4	looseUnmarshal	1336
6.258.3.5	tightMarshal1	1336
6.258.3.6	tightMarshal2	1337
6.258.3.7	tightUnmarshal	1337
6.259	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller Class Reference	1338
6.259.1	Detailed Description	1339
6.259.2	Constructor & Destructor Documentation	1339
6.259.2.1	ConsumerIdMarshaller	1339
6.259.2.2	~ConsumerIdMarshaller	1339
6.259.3	Member Function Documentation	1339
6.259.3.1	createObject	1339
6.259.3.2	getDataStructureType	1339
6.259.3.3	looseMarshal	1340

6.259.3.4 looseUnmarshal	1340
6.259.3.5 tightMarshal1	1340
6.259.3.6 tightMarshal2	1341
6.259.3.7 tightUnmarshal	1341
6.260activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller Class Reference	1342
6.260.1 Detailed Description	1343
6.260.2 Constructor & Destructor Documentation	1343
6.260.2.1 ConsumerIdMarshaller	1343
6.260.2.2 ~ConsumerIdMarshaller	1343
6.260.3 Member Function Documentation	1343
6.260.3.1 createObject	1343
6.260.3.2 getDataStructureType	1343
6.260.3.3 looseMarshal	1343
6.260.3.4 looseUnmarshal	1344
6.260.3.5 tightMarshal1	1344
6.260.3.6 tightMarshal2	1345
6.260.3.7 tightUnmarshal	1345
6.261activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller Class Reference	1345
6.261.1 Detailed Description	1346
6.261.2 Constructor & Destructor Documentation	1347
6.261.2.1 ConsumerIdMarshaller	1347
6.261.2.2 ~ConsumerIdMarshaller	1347
6.261.3 Member Function Documentation	1347
6.261.3.1 createObject	1347
6.261.3.2 getDataStructureType	1347
6.261.3.3 looseMarshal	1347
6.261.3.4 looseUnmarshal	1348
6.261.3.5 tightMarshal1	1348
6.261.3.6 tightMarshal2	1348
6.261.3.7 tightUnmarshal	1349
6.262activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller Class Reference	1349
6.262.1 Detailed Description	1350
6.262.2 Constructor & Destructor Documentation	1351
6.262.2.1 ConsumerIdMarshaller	1351
6.262.2.2 ~ConsumerIdMarshaller	1351
6.262.3 Member Function Documentation	1351

6.262.3.1	createObject	1351
6.262.3.2	getDataStructureType	1351
6.262.3.3	looseMarshal	1351
6.262.3.4	looseUnmarshal	1352
6.262.3.5	tightMarshal1	1352
6.262.3.6	tightMarshal2	1352
6.262.3.7	tightUnmarshal	1353
6.263	activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller Class Reference	1353
6.263.1	Detailed Description	1354
6.263.2	Constructor & Destructor Documentation	1355
6.263.2.1	ConsumerIdMarshaller	1355
6.263.2.2	~ConsumerIdMarshaller	1355
6.263.3	Member Function Documentation	1355
6.263.3.1	createObject	1355
6.263.3.2	getDataStructureType	1355
6.263.3.3	looseMarshal	1355
6.263.3.4	looseUnmarshal	1356
6.263.3.5	tightMarshal1	1356
6.263.3.6	tightMarshal2	1356
6.263.3.7	tightUnmarshal	1357
6.264	activemq::commands::ConsumerInfo Class Reference	1357
6.264.1	Constructor & Destructor Documentation	1360
6.264.1.1	ConsumerInfo	1360
6.264.1.2	~ConsumerInfo	1360
6.264.2	Member Function Documentation	1360
6.264.2.1	cloneDataStructure	1360
6.264.2.2	copyDataStructure	1360
6.264.2.3	createRemoveCommand	1360
6.264.2.4	equals	1360
6.264.2.5	getAdditionalPredicate	1361
6.264.2.6	getAdditionalPredicate	1361
6.264.2.7	getBrokerPath	1361
6.264.2.8	getBrokerPath	1361
6.264.2.9	getConsumerId	1361
6.264.2.10	getConsumerId	1361
6.264.2.11	getDataStructureType	1361

6.264.2.12	getDestination	1362
6.264.2.13	getDestination	1362
6.264.2.14	getMaximumPendingMessageLimit	1362
6.264.2.15	getNetworkConsumerPath	1362
6.264.2.16	getNetworkConsumerPath	1362
6.264.2.17	getPrefetchSize	1362
6.264.2.18	getPriority	1362
6.264.2.19	getSelector	1362
6.264.2.20	getSelector	1362
6.264.2.21	getSubscriptionName	1362
6.264.2.22	getSubscriptionName	1362
6.264.2.23	sBrowser	1362
6.264.2.24	sConsumerInfo	1362
6.264.2.25	sDispatchAsync	1363
6.264.2.26	sExclusive	1363
6.264.2.27	sNetworkSubscription	1363
6.264.2.28	sNoLocal	1363
6.264.2.29	sNoRangeAcks	1363
6.264.2.30	sOptimizedAcknowledge	1363
6.264.2.31	isRetroactive	1363
6.264.2.32	setAdditionalPredicate	1363
6.264.2.33	setBrokerPath	1363
6.264.2.34	setBrowser	1363
6.264.2.35	setConsumerId	1363
6.264.2.36	setDestination	1363
6.264.2.37	setDispatchAsync	1363
6.264.2.38	setExclusive	1363
6.264.2.39	setMaximumPendingMessageLimit	1363
6.264.2.40	setNetworkConsumerPath	1363
6.264.2.41	setNetworkSubscription	1363
6.264.2.42	setNoLocal	1363
6.264.2.43	setNoRangeAcks	1363
6.264.2.44	setOptimizedAcknowledge	1363
6.264.2.45	setPrefetchSize	1363
6.264.2.46	setPriority	1363
6.264.2.47	setRetroactive	1363

6.264.2.4	setSelector	1363
6.264.2.4	setSubscriptionName	1363
6.264.2.5	toString	1363
6.264.2.5	visit	1364
6.264.3	Field Documentation	1365
6.264.3.1	additionalPredicate	1365
6.264.3.2	brokerPath	1365
6.264.3.3	browser	1365
6.264.3.4	consumerId	1365
6.264.3.5	destination	1365
6.264.3.6	dispatchAsync	1365
6.264.3.7	exclusive	1365
6.264.3.8	ID_CONSUMERINFO	1365
6.264.3.9	maximumPendingMessageLimit	1365
6.264.3.10	networkConsumerPath	1365
6.264.3.11	networkSubscription	1365
6.264.3.12	noLocal	1365
6.264.3.13	noRangeAcks	1365
6.264.3.14	optimizedAcknowledge	1365
6.264.3.15	prefetchSize	1365
6.264.3.16	priority	1365
6.264.3.17	retroactive	1365
6.264.3.18	selector	1365
6.264.3.19	subscriptionName	1365
6.265	activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller Class Reference	1366
6.265.1	Detailed Description	1367
6.265.2	Constructor & Destructor Documentation	1367
6.265.2.1	ConsumerInfoMarshaller	1367
6.265.2.2	~ConsumerInfoMarshaller	1367
6.265.3	Member Function Documentation	1367
6.265.3.1	createObject	1367
6.265.3.2	getDataStructureType	1367
6.265.3.3	looseMarshal	1367
6.265.3.4	looseUnmarshal	1368
6.265.3.5	tightMarshal1	1368
6.265.3.6	tightMarshal2	1369

6.265.3.7 tightUnmarshal	1369
6.266activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller Class Reference	1369
6.266.1 Detailed Description	1370
6.266.2 Constructor & Destructor Documentation	1371
6.266.2.1 ConsumerInfoMarshaller	1371
6.266.2.2 ~ConsumerInfoMarshaller	1371
6.266.3 Member Function Documentation	1371
6.266.3.1 createObject	1371
6.266.3.2 getDataStructureType	1371
6.266.3.3 looseMarshal	1371
6.266.3.4 looseUnmarshal	1372
6.266.3.5 tightMarshal1	1372
6.266.3.6 tightMarshal2	1372
6.266.3.7 tightUnmarshal	1373
6.267activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller Class Reference	1373
6.267.1 Detailed Description	1374
6.267.2 Constructor & Destructor Documentation	1375
6.267.2.1 ConsumerInfoMarshaller	1375
6.267.2.2 ~ConsumerInfoMarshaller	1375
6.267.3 Member Function Documentation	1375
6.267.3.1 createObject	1375
6.267.3.2 getDataStructureType	1375
6.267.3.3 looseMarshal	1375
6.267.3.4 looseUnmarshal	1376
6.267.3.5 tightMarshal1	1376
6.267.3.6 tightMarshal2	1376
6.267.3.7 tightUnmarshal	1377
6.268activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller Class Reference	1377
6.268.1 Detailed Description	1378
6.268.2 Constructor & Destructor Documentation	1379
6.268.2.1 ConsumerInfoMarshaller	1379
6.268.2.2 ~ConsumerInfoMarshaller	1379
6.268.3 Member Function Documentation	1379
6.268.3.1 createObject	1379

6.268.3.2	getDataStructureType	1379
6.268.3.3	looseMarshal	1379
6.268.3.4	looseUnmarshal	1380
6.268.3.5	tightMarshal1	1380
6.268.3.6	tightMarshal2	1380
6.268.3.7	tightUnmarshal	1381
6.269	activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller Class Reference	1381
6.269.1	Detailed Description	1382
6.269.2	Constructor & Destructor Documentation	1383
6.269.2.1	ConsumerInfoMarshaller	1383
6.269.2.2	~ConsumerInfoMarshaller	1383
6.269.3	Member Function Documentation	1383
6.269.3.1	createObject	1383
6.269.3.2	getDataStructureType	1383
6.269.3.3	looseMarshal	1383
6.269.3.4	looseUnmarshal	1384
6.269.3.5	tightMarshal1	1384
6.269.3.6	tightMarshal2	1384
6.269.3.7	tightUnmarshal	1385
6.270	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller Class Reference	1385
6.270.1	Detailed Description	1386
6.270.2	Constructor & Destructor Documentation	1387
6.270.2.1	ConsumerInfoMarshaller	1387
6.270.2.2	~ConsumerInfoMarshaller	1387
6.270.3	Member Function Documentation	1387
6.270.3.1	createObject	1387
6.270.3.2	getDataStructureType	1387
6.270.3.3	looseMarshal	1387
6.270.3.4	looseUnmarshal	1388
6.270.3.5	tightMarshal1	1388
6.270.3.6	tightMarshal2	1388
6.270.3.7	tightUnmarshal	1389
6.271	activemq::state::ConsumerState Class Reference	1389
6.271.1	Constructor & Destructor Documentation	1390
6.271.1.1	ConsumerState	1390

6.271.1.2 <code>~ConsumerState</code>	1390
6.271.2 Member Function Documentation	1390
6.271.2.1 <code>getInfo</code>	1390
6.271.2.2 <code>toString</code>	1390
6.272 <code>activemq::commands::ControlCommand</code> Class Reference	1390
6.272.1 Constructor & Destructor Documentation	1391
6.272.1.1 <code>ControlCommand</code>	1391
6.272.1.2 <code>~ControlCommand</code>	1391
6.272.2 Member Function Documentation	1391
6.272.2.1 <code>cloneDataStructure</code>	1391
6.272.2.2 <code>copyDataStructure</code>	1391
6.272.2.3 <code>equals</code>	1392
6.272.2.4 <code>getCommand</code>	1392
6.272.2.5 <code>getCommand</code>	1392
6.272.2.6 <code>getDataStructureType</code>	1392
6.272.2.7 <code>setCommand</code>	1392
6.272.2.8 <code>toString</code>	1392
6.272.2.9 <code>visit</code>	1393
6.272.3 Field Documentation	1393
6.272.3.1 <code>command</code>	1393
6.272.3.2 <code>ID_CONTROLCOMMAND</code>	1393
6.273 <code>activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller</code> Class Reference	1393
6.273.1 Detailed Description	1394
6.273.2 Constructor & Destructor Documentation	1394
6.273.2.1 <code>ControlCommandMarshaller</code>	1394
6.273.2.2 <code>~ControlCommandMarshaller</code>	1394
6.273.3 Member Function Documentation	1394
6.273.3.1 <code>createObject</code>	1394
6.273.3.2 <code>getDataStructureType</code>	1395
6.273.3.3 <code>looseMarshal</code>	1395
6.273.3.4 <code>looseUnmarshal</code>	1395
6.273.3.5 <code>tightMarshal1</code>	1396
6.273.3.6 <code>tightMarshal2</code>	1396
6.273.3.7 <code>tightUnmarshal</code>	1396
6.274 <code>activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller</code> Class Reference	1397

6.274.1 Detailed Description	1398
6.274.2 Constructor & Destructor Documentation	1398
6.274.2.1 ControlCommandMarshaller	1398
6.274.2.2 ~ControlCommandMarshaller	1398
6.274.3 Member Function Documentation	1398
6.274.3.1 createObject	1398
6.274.3.2 getDataStructureType	1398
6.274.3.3 looseMarshal	1399
6.274.3.4 looseUnmarshal	1399
6.274.3.5 tightMarshal1	1399
6.274.3.6 tightMarshal2	1400
6.274.3.7 tightUnmarshal	1400
6.275activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller Class	
Reference	1401
6.275.1 Detailed Description	1402
6.275.2 Constructor & Destructor Documentation	1402
6.275.2.1 ControlCommandMarshaller	1402
6.275.2.2 ~ControlCommandMarshaller	1402
6.275.3 Member Function Documentation	1402
6.275.3.1 createObject	1402
6.275.3.2 getDataStructureType	1402
6.275.3.3 looseMarshal	1403
6.275.3.4 looseUnmarshal	1403
6.275.3.5 tightMarshal1	1403
6.275.3.6 tightMarshal2	1404
6.275.3.7 tightUnmarshal	1404
6.276activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller Class	
Reference	1405
6.276.1 Detailed Description	1406
6.276.2 Constructor & Destructor Documentation	1406
6.276.2.1 ControlCommandMarshaller	1406
6.276.2.2 ~ControlCommandMarshaller	1406
6.276.3 Member Function Documentation	1406
6.276.3.1 createObject	1406
6.276.3.2 getDataStructureType	1406
6.276.3.3 looseMarshal	1407
6.276.3.4 looseUnmarshal	1407

6.276.3.5 tightMarshal1	1407
6.276.3.6 tightMarshal2	1408
6.276.3.7 tightUnmarshal	1408
6.277activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller Class	
Reference	1409
6.277.1 Detailed Description	1410
6.277.2 Constructor & Destructor Documentation	1410
6.277.2.1 ControlCommandMarshaller	1410
6.277.2.2 ~ControlCommandMarshaller	1410
6.277.3 Member Function Documentation	1410
6.277.3.1 createObject	1410
6.277.3.2 getDataStructureType	1410
6.277.3.3 looseMarshal	1411
6.277.3.4 looseUnmarshal	1411
6.277.3.5 tightMarshal1	1411
6.277.3.6 tightMarshal2	1412
6.277.3.7 tightUnmarshal	1412
6.278activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller Class	
Reference	1413
6.278.1 Detailed Description	1414
6.278.2 Constructor & Destructor Documentation	1414
6.278.2.1 ControlCommandMarshaller	1414
6.278.2.2 ~ControlCommandMarshaller	1414
6.278.3 Member Function Documentation	1414
6.278.3.1 createObject	1414
6.278.3.2 getDataStructureType	1414
6.278.3.3 looseMarshal	1415
6.278.3.4 looseUnmarshal	1415
6.278.3.5 tightMarshal1	1415
6.278.3.6 tightMarshal2	1416
6.278.3.7 tightUnmarshal	1416
6.279decaf::util::concurrent::CountDownLatch Class Reference	1417
6.279.1 Constructor & Destructor Documentation	1417
6.279.1.1 CountDownLatch	1417
6.279.1.2 ~CountDownLatch	1418
6.279.2 Member Function Documentation	1418
6.279.2.1 await	1418

6.279.2.2 await	1418
6.279.2.3 await	1419
6.279.2.4 countDown	1419
6.279.2.5 getCount	1419
6.280decaf::util::zip::CRC32 Class Reference	1420
6.280.1 Detailed Description	1420
6.280.2 Constructor & Destructor Documentation	1421
6.280.2.1 CRC32	1421
6.280.2.2 ~CRC32	1421
6.280.3 Member Function Documentation	1421
6.280.3.1 getValue	1421
6.280.3.2 reset	1421
6.280.3.3 update	1421
6.280.3.4 update	1421
6.280.3.5 update	1422
6.280.3.6 update	1422
6.281ct_data_s Struct Reference	1422
6.281.1 Field Documentation	1423
6.281.1.1 code	1423
6.281.1.2 dad	1423
6.281.1.3 dl	1423
6.281.1.4 fc	1423
6.281.1.5 freq	1423
6.281.1.6 len	1423
6.282activemq::commands::DataArrayResponse Class Reference	1423
6.282.1 Constructor & Destructor Documentation	1424
6.282.1.1 DataArrayResponse	1424
6.282.1.2 ~DataArrayResponse	1424
6.282.2 Member Function Documentation	1424
6.282.2.1 cloneDataStructure	1424
6.282.2.2 copyDataStructure	1424
6.282.2.3 equals	1424
6.282.2.4 getData	1425
6.282.2.5 getData	1425
6.282.2.6 getDataStructureType	1425
6.282.2.7 setData	1425

6.282.2.8 toString	1425
6.282.3 Field Documentation	1425
6.282.3.1 data	1425
6.282.3.2 ID_DATAARRAYRESPONSE	1425
6.283activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller Class	
Reference	1426
6.283.1 Detailed Description	1427
6.283.2 Constructor & Destructor Documentation	1427
6.283.2.1 DataArrayResponseMarshaller	1427
6.283.2.2 ~DataArrayResponseMarshaller	1427
6.283.3 Member Function Documentation	1427
6.283.3.1 createObject	1427
6.283.3.2 getDataStructureType	1427
6.283.3.3 looseMarshal	1427
6.283.3.4 looseUnmarshal	1428
6.283.3.5 tightMarshal1	1428
6.283.3.6 tightMarshal2	1429
6.283.3.7 tightUnmarshal	1429
6.284activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller Class	
Reference	1430
6.284.1 Detailed Description	1431
6.284.2 Constructor & Destructor Documentation	1431
6.284.2.1 DataArrayResponseMarshaller	1431
6.284.2.2 ~DataArrayResponseMarshaller	1431
6.284.3 Member Function Documentation	1431
6.284.3.1 createObject	1431
6.284.3.2 getDataStructureType	1431
6.284.3.3 looseMarshal	1431
6.284.3.4 looseUnmarshal	1432
6.284.3.5 tightMarshal1	1432
6.284.3.6 tightMarshal2	1433
6.284.3.7 tightUnmarshal	1433
6.285activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller Class	
Reference	1434
6.285.1 Detailed Description	1435
6.285.2 Constructor & Destructor Documentation	1435
6.285.2.1 DataArrayResponseMarshaller	1435

6.285.2.2 ~DataArrayResponseMarshaller	1435
6.285.3 Member Function Documentation	1435
6.285.3.1 createObject	1435
6.285.3.2 getDataStructureType	1435
6.285.3.3 looseMarshal	1435
6.285.3.4 looseUnmarshal	1436
6.285.3.5 tightMarshal1	1436
6.285.3.6 tightMarshal2	1437
6.285.3.7 tightUnmarshal	1437
6.286activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller Class	
Reference	1438
6.286.1 Detailed Description	1439
6.286.2 Constructor & Destructor Documentation	1439
6.286.2.1 DataArrayResponseMarshaller	1439
6.286.2.2 ~DataArrayResponseMarshaller	1439
6.286.3 Member Function Documentation	1439
6.286.3.1 createObject	1439
6.286.3.2 getDataStructureType	1439
6.286.3.3 looseMarshal	1439
6.286.3.4 looseUnmarshal	1440
6.286.3.5 tightMarshal1	1440
6.286.3.6 tightMarshal2	1441
6.286.3.7 tightUnmarshal	1441
6.287activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller Class	
Reference	1442
6.287.1 Detailed Description	1443
6.287.2 Constructor & Destructor Documentation	1443
6.287.2.1 DataArrayResponseMarshaller	1443
6.287.2.2 ~DataArrayResponseMarshaller	1443
6.287.3 Member Function Documentation	1443
6.287.3.1 createObject	1443
6.287.3.2 getDataStructureType	1443
6.287.3.3 looseMarshal	1443
6.287.3.4 looseUnmarshal	1444
6.287.3.5 tightMarshal1	1444
6.287.3.6 tightMarshal2	1445
6.287.3.7 tightUnmarshal	1445

6.288	activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller Class Reference	1446
6.288.1	Detailed Description	1447
6.288.2	Constructor & Destructor Documentation	1447
6.288.2.1	DataArrayResponseMarshaller	1447
6.288.2.2	~DataArrayResponseMarshaller	1447
6.288.3	Member Function Documentation	1447
6.288.3.1	createObject	1447
6.288.3.2	getDataStructureType	1447
6.288.3.3	looseMarshal	1447
6.288.3.4	looseUnmarshal	1448
6.288.3.5	tightMarshal1	1448
6.288.3.6	tightMarshal2	1449
6.288.3.7	tightUnmarshal	1449
6.289	decaf::util::zip::DataFormatException Class Reference	1450
6.289.1	Constructor & Destructor Documentation	1450
6.289.1.1	DataFormatException	1450
6.289.1.2	DataFormatException	1450
6.289.1.3	DataFormatException	1451
6.289.1.4	DataFormatException	1451
6.289.1.5	DataFormatException	1451
6.289.1.6	DataFormatException	1451
6.289.1.7	~DataFormatException	1452
6.289.2	Member Function Documentation	1452
6.289.2.1	clone	1452
6.290	decaf::io::DataInput Class Reference	1452
6.290.1	Detailed Description	1453
6.290.2	Constructor & Destructor Documentation	1454
6.290.2.1	~DataInput	1454
6.290.3	Member Function Documentation	1454
6.290.3.1	readBoolean	1454
6.290.3.2	readByte	1454
6.290.3.3	readChar	1455
6.290.3.4	readDouble	1455
6.290.3.5	readFloat	1455
6.290.3.6	readFully	1456
6.290.3.7	readFully	1456

6.290.3.8 readInt	1457
6.290.3.9 readLine	1457
6.290.3.10 readLong	1458
6.290.3.11 readShort	1458
6.290.3.12 readString	1458
6.290.3.13 readUnsignedByte	1459
6.290.3.14 readUnsignedShort	1459
6.290.3.15 readUTF	1459
6.290.3.16 skipBytes	1460
6.291 decaf::io::DataInputStream Class Reference	1460
6.291.1 Detailed Description	1462
6.291.2 Constructor & Destructor Documentation	1462
6.291.2.1 DataInputStream	1462
6.291.2.2 ~DataInputStream	1462
6.291.3 Member Function Documentation	1462
6.291.3.1 readBoolean	1462
6.291.3.2 readByte	1462
6.291.3.3 readChar	1463
6.291.3.4 readDouble	1463
6.291.3.5 readFloat	1463
6.291.3.6 readFully	1464
6.291.3.7 readFully	1464
6.291.3.8 readInt	1465
6.291.3.9 readLine	1465
6.291.3.10 readLong	1466
6.291.3.11 readShort	1466
6.291.3.12 readString	1466
6.291.3.13 readUnsignedByte	1467
6.291.3.14 readUnsignedShort	1467
6.291.3.15 readUTF	1467
6.291.3.16 skipBytes	1468
6.292 decaf::io::DataOutput Class Reference	1468
6.292.1 Detailed Description	1469
6.292.2 Constructor & Destructor Documentation	1470
6.292.2.1 ~DataOutput	1470
6.292.3 Member Function Documentation	1470

6.292.3.1 writeBoolean	1470
6.292.3.2 writeByte	1470
6.292.3.3 writeBytes	1470
6.292.3.4 writeChar	1471
6.292.3.5 writeChars	1471
6.292.3.6 writeDouble	1471
6.292.3.7 writeFloat	1471
6.292.3.8 writeInt	1472
6.292.3.9 writeLong	1472
6.292.3.10 writeShort	1472
6.292.3.11 writeUnsignedShort	1473
6.292.3.12 writeUTF	1473
6.293 decaf::io::DataOutputStream Class Reference	1473
6.293.1 Detailed Description	1475
6.293.2 Constructor & Destructor Documentation	1475
6.293.2.1 DataOutputStream	1475
6.293.2.2 ~DataOutputStream	1475
6.293.3 Member Function Documentation	1475
6.293.3.1 doWriteArrayBounded	1475
6.293.3.2 doWriteByte	1475
6.293.3.3 size	1475
6.293.3.4 writeBoolean	1476
6.293.3.5 writeByte	1476
6.293.3.6 writeBytes	1476
6.293.3.7 writeChar	1476
6.293.3.8 writeChars	1476
6.293.3.9 writeDouble	1476
6.293.3.10 writeFloat	1476
6.293.3.11 writeInt	1476
6.293.3.12 writeLong	1476
6.293.3.13 writeShort	1476
6.293.3.14 writeUnsignedShort	1476
6.293.3.15 writeUTF	1476
6.293.4 Field Documentation	1476
6.293.4.1 buffer	1476
6.293.4.2 written	1476

6.294activemq::commands::DataResponse Class Reference	1476
6.294.1 Constructor & Destructor Documentation	1478
6.294.1.1 DataResponse	1478
6.294.1.2 ~DataResponse	1478
6.294.2 Member Function Documentation	1478
6.294.2.1 cloneDataStructure	1478
6.294.2.2 copyDataStructure	1478
6.294.2.3 equals	1478
6.294.2.4 getData	1479
6.294.2.5 getData	1479
6.294.2.6 getDataStructureType	1479
6.294.2.7 setData	1479
6.294.2.8 toString	1479
6.294.3 Field Documentation	1479
6.294.3.1 data	1479
6.294.3.2 ID_DATARESPONSE	1479
6.295activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller Class Refer- ence	1479
6.295.1 Detailed Description	1480
6.295.2 Constructor & Destructor Documentation	1481
6.295.2.1 DataResponseMarshaller	1481
6.295.2.2 ~DataResponseMarshaller	1481
6.295.3 Member Function Documentation	1481
6.295.3.1 createObject	1481
6.295.3.2 getDataStructureType	1481
6.295.3.3 looseMarshal	1481
6.295.3.4 looseUnmarshal	1482
6.295.3.5 tightMarshal1	1482
6.295.3.6 tightMarshal2	1483
6.295.3.7 tightUnmarshal	1483
6.296activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller Class Refer- ence	1483
6.296.1 Detailed Description	1484
6.296.2 Constructor & Destructor Documentation	1485
6.296.2.1 DataResponseMarshaller	1485
6.296.2.2 ~DataResponseMarshaller	1485
6.296.3 Member Function Documentation	1485

6.296.3.1 createObject	1485
6.296.3.2 getDataStructureType	1485
6.296.3.3 looseMarshal	1485
6.296.3.4 looseUnmarshal	1486
6.296.3.5 tightMarshal1	1486
6.296.3.6 tightMarshal2	1487
6.296.3.7 tightUnmarshal	1487
6.297activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller Class Reference	1487
6.297.1 Detailed Description	1488
6.297.2 Constructor & Destructor Documentation	1489
6.297.2.1 DataResponseMarshaller	1489
6.297.2.2 ~DataResponseMarshaller	1489
6.297.3 Member Function Documentation	1489
6.297.3.1 createObject	1489
6.297.3.2 getDataStructureType	1489
6.297.3.3 looseMarshal	1489
6.297.3.4 looseUnmarshal	1490
6.297.3.5 tightMarshal1	1490
6.297.3.6 tightMarshal2	1491
6.297.3.7 tightUnmarshal	1491
6.298activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller Class Reference	1491
6.298.1 Detailed Description	1492
6.298.2 Constructor & Destructor Documentation	1493
6.298.2.1 DataResponseMarshaller	1493
6.298.2.2 ~DataResponseMarshaller	1493
6.298.3 Member Function Documentation	1493
6.298.3.1 createObject	1493
6.298.3.2 getDataStructureType	1493
6.298.3.3 looseMarshal	1493
6.298.3.4 looseUnmarshal	1494
6.298.3.5 tightMarshal1	1494
6.298.3.6 tightMarshal2	1495
6.298.3.7 tightUnmarshal	1495
6.299activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller Class Reference	1495

6.299.1 Detailed Description	1496
6.299.2 Constructor & Destructor Documentation	1497
6.299.2.1 DataResponseMarshaller	1497
6.299.2.2 ~DataResponseMarshaller	1497
6.299.3 Member Function Documentation	1497
6.299.3.1 createObject	1497
6.299.3.2 getDataStructureType	1497
6.299.3.3 looseMarshal	1497
6.299.3.4 looseUnmarshal	1498
6.299.3.5 tightMarshal1	1498
6.299.3.6 tightMarshal2	1499
6.299.3.7 tightUnmarshal	1499
6.300activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller Class Reference	1499
6.300.1 Detailed Description	1500
6.300.2 Constructor & Destructor Documentation	1501
6.300.2.1 DataResponseMarshaller	1501
6.300.2.2 ~DataResponseMarshaller	1501
6.300.3 Member Function Documentation	1501
6.300.3.1 createObject	1501
6.300.3.2 getDataStructureType	1501
6.300.3.3 looseMarshal	1501
6.300.3.4 looseUnmarshal	1502
6.300.3.5 tightMarshal1	1502
6.300.3.6 tightMarshal2	1503
6.300.3.7 tightUnmarshal	1503
6.301activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference	1503
6.301.1 Detailed Description	1504
6.301.2 Constructor & Destructor Documentation	1505
6.301.2.1 ~DataStreamMarshaller	1505
6.301.3 Member Function Documentation	1505
6.301.3.1 createObject	1505
6.301.3.2 getDataStructureType	1511
6.301.3.3 looseMarshal	1518
6.301.3.4 looseUnmarshal	1525
6.301.3.5 tightMarshal1	1532
6.301.3.6 tightMarshal2	1539

6.301.3.7 tightUnmarshal	1546
6.302activemq::commands::DataStructure Class Reference	1553
6.302.1 Constructor & Destructor Documentation	1554
6.302.1.1 ~DataStructure	1554
6.302.2 Member Function Documentation	1554
6.302.2.1 cloneDataStructure	1554
6.302.2.2 copyDataStructure	1555
6.302.2.3 equals	1556
6.302.2.4 getDataStructureType	1557
6.302.2.5 toString	1558
6.303decaf::util::Date Class Reference	1559
6.303.1 Detailed Description	1560
6.303.2 Constructor & Destructor Documentation	1560
6.303.2.1 Date	1560
6.303.2.2 Date	1560
6.303.2.3 Date	1560
6.303.2.4 ~Date	1560
6.303.3 Member Function Documentation	1560
6.303.3.1 after	1560
6.303.3.2 before	1561
6.303.3.3 compareTo	1561
6.303.3.4 equals	1561
6.303.3.5 getTime	1561
6.303.3.6 operator<	1561
6.303.3.7 operator=	1562
6.303.3.8 operator==	1562
6.303.3.9 setTime	1562
6.303.3.10 toString	1562
6.304decaf::internal::DecafRuntime Class Reference	1563
6.304.1 Detailed Description	1563
6.304.2 Constructor & Destructor Documentation	1563
6.304.2.1 DecafRuntime	1563
6.304.2.2 ~DecafRuntime	1564
6.304.3 Member Function Documentation	1564
6.304.3.1 getGlobalPool	1564
6.305activemq::threads::DedicatedTaskRunner Class Reference	1564

6.305.1 Constructor & Destructor Documentation	1565
6.305.1.1 DedicatedTaskRunner	1565
6.305.1.2 ~DedicatedTaskRunner	1565
6.305.2 Member Function Documentation	1565
6.305.2.1 run	1565
6.305.2.2 shutdown	1565
6.305.2.3 shutdown	1565
6.305.2.4 wakeup	1565
6.306activemq::core::policies::DefaultPrefetchPolicy Class Reference	1565
6.306.1 Constructor & Destructor Documentation	1567
6.306.1.1 DefaultPrefetchPolicy	1567
6.306.1.2 ~DefaultPrefetchPolicy	1567
6.306.2 Member Function Documentation	1567
6.306.2.1 clone	1567
6.306.2.2 getDurableTopicPrefetch	1567
6.306.2.3 getMaxPrefetchLimit	1567
6.306.2.4 getQueueBrowserPrefetch	1567
6.306.2.5 getQueuePrefetch	1568
6.306.2.6 getTopicPrefetch	1568
6.306.2.7 setDurableTopicPrefetch	1568
6.306.2.8 setQueueBrowserPrefetch	1568
6.306.2.9 setQueuePrefetch	1569
6.306.2.10setTopicPrefetch	1569
6.306.3 Field Documentation	1569
6.306.3.1 DEFAULT_DURABLE_TOPIC_PREFETCH	1569
6.306.3.2 DEFAULT_QUEUE_BROWSER_PREFETCH	1569
6.306.3.3 DEFAULT_QUEUE_PREFETCH	1569
6.306.3.4 DEFAULT_TOPIC_PREFETCH	1569
6.306.3.5 MAX_PREFETCH_SIZE	1569
6.307activemq::core::policies::DefaultRedeliveryPolicy Class Reference	1569
6.307.1 Constructor & Destructor Documentation	1570
6.307.1.1 DefaultRedeliveryPolicy	1570
6.307.1.2 ~DefaultRedeliveryPolicy	1570
6.307.2 Member Function Documentation	1570
6.307.2.1 clone	1570
6.307.2.2 getBackOffMultiplier	1571

6.307.2.3	getCollisionAvoidancePercent	1571
6.307.2.4	getInitialRedeliveryDelay	1571
6.307.2.5	getMaximumRedeliveries	1571
6.307.2.6	getRedeliveryDelay	1572
6.307.2.7	isUseCollisionAvoidance	1572
6.307.2.8	isUseExponentialBackOff	1572
6.307.2.9	setBackOffMultiplier	1572
6.307.2.10	setCollisionAvoidancePercent	1572
6.307.2.11	setInitialRedeliveryDelay	1573
6.307.2.12	setMaximumRedeliveries	1573
6.307.2.13	setUseCollisionAvoidance	1573
6.307.2.14	setUseExponentialBackOff	1573
6.308	decaf::internal::net::DefaultServerSocketFactory Class Reference	1574
6.308.1	Detailed Description	1575
6.308.2	Constructor & Destructor Documentation	1575
6.308.2.1	DefaultServerSocketFactory	1575
6.308.2.2	~DefaultServerSocketFactory	1575
6.308.3	Member Function Documentation	1575
6.308.3.1	createServerSocket	1575
6.308.3.2	createServerSocket	1576
6.308.3.3	createServerSocket	1576
6.308.3.4	createServerSocket	1576
6.309	decaf::internal::net::DefaultSocketFactory Class Reference	1577
6.309.1	Detailed Description	1578
6.309.2	Constructor & Destructor Documentation	1579
6.309.2.1	DefaultSocketFactory	1579
6.309.2.2	~DefaultSocketFactory	1579
6.309.3	Member Function Documentation	1579
6.309.3.1	createSocket	1579
6.309.3.2	createSocket	1579
6.309.3.3	createSocket	1580
6.309.3.4	createSocket	1580
6.309.3.5	createSocket	1581
6.310	decaf::internal::net::ssl::DefaultSSLContext Class Reference	1581
6.310.1	Detailed Description	1582
6.310.2	Constructor & Destructor Documentation	1582

6.310.2.1 DefaultSSLContext	1582
6.310.2.2 ~DefaultSSLContext	1582
6.310.3 Member Function Documentation	1582
6.310.3.1 getContext	1582
6.311decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference	1582
6.311.1 Detailed Description	1584
6.311.2 Constructor & Destructor Documentation	1584
6.311.2.1 DefaultSSLServerSocketFactory	1584
6.311.2.2 ~DefaultSSLServerSocketFactory	1584
6.311.3 Member Function Documentation	1584
6.311.3.1 createServerSocket	1584
6.311.3.2 createServerSocket	1585
6.311.3.3 createServerSocket	1585
6.311.3.4 createServerSocket	1585
6.311.3.5 getDefaultCipherSuites	1586
6.311.3.6 getSupportedCipherSuites	1586
6.312decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference	1587
6.312.1 Detailed Description	1589
6.312.2 Constructor & Destructor Documentation	1589
6.312.2.1 DefaultSSLSocketFactory	1589
6.312.2.2 ~DefaultSSLSocketFactory	1589
6.312.3 Member Function Documentation	1589
6.312.3.1 createSocket	1589
6.312.3.2 createSocket	1590
6.312.3.3 createSocket	1590
6.312.3.4 createSocket	1591
6.312.3.5 createSocket	1591
6.312.3.6 createSocket	1592
6.312.3.7 getDefaultCipherSuites	1592
6.312.3.8 getSupportedCipherSuites	1592
6.313activemq::transport::DefaultTransportListener Class Reference	1593
6.313.1 Constructor & Destructor Documentation	1594
6.313.1.1 ~DefaultTransportListener	1594
6.313.2 Member Function Documentation	1594
6.313.2.1 onCommand	1594
6.313.2.2 onException	1594

6.313.2.3 transportInterrupted	1594
6.313.2.4 transportResumed	1594
6.314decaf::util::zip::Deflater Class Reference	1595
6.314.1 Detailed Description	1597
6.314.2 Constructor & Destructor Documentation	1597
6.314.2.1 Deflater	1597
6.314.2.2 Deflater	1597
6.314.2.3 ~Deflater	1598
6.314.3 Member Function Documentation	1598
6.314.3.1 deflate	1598
6.314.3.2 deflate	1598
6.314.3.3 deflate	1599
6.314.3.4 end	1599
6.314.3.5 finish	1599
6.314.3.6 finished	1599
6.314.3.7 getAdler	1599
6.314.3.8 getBytesRead	1600
6.314.3.9 getBytesWritten	1600
6.314.3.10needsInput	1600
6.314.3.11reset	1600
6.314.3.12setDictionary	1600
6.314.3.13setDictionary	1601
6.314.3.14setDictionary	1601
6.314.3.15setInput	1602
6.314.3.16setInput	1602
6.314.3.17setInput	1602
6.314.3.18setLevel	1603
6.314.3.19setStrategy	1603
6.314.4 Field Documentation	1603
6.314.4.1 BEST_COMPRESSION	1603
6.314.4.2 BEST_SPEED	1603
6.314.4.3 DEFAULT_COMPRESSION	1604
6.314.4.4 DEFAULT_STRATEGY	1604
6.314.4.5 DEFLATED	1604
6.314.4.6 FILTERED	1604
6.314.4.7 HUFFMAN_ONLY	1604

6.314.4.8 NO_COMPRESSION	1604
6.315decaf::util::zip::DeflaterOutputStream Class Reference	1604
6.315.1 Detailed Description	1606
6.315.2 Constructor & Destructor Documentation	1606
6.315.2.1 DeflaterOutputStream	1606
6.315.2.2 DeflaterOutputStream	1606
6.315.2.3 DeflaterOutputStream	1606
6.315.2.4 ~DeflaterOutputStream	1607
6.315.3 Member Function Documentation	1607
6.315.3.1 close	1607
6.315.3.2 deflate	1607
6.315.3.3 doWriteArrayBounded	1607
6.315.3.4 doWriteByte	1607
6.315.3.5 finish	1608
6.315.4 Field Documentation	1608
6.315.4.1 buf	1608
6.315.4.2 DEFAULT_BUFFER_SIZE	1608
6.315.4.3 deflater	1608
6.315.4.4 isDone	1608
6.315.4.5 ownDeflater	1608
6.316decaf::util::concurrent::Delayed Class Reference	1608
6.316.1 Detailed Description	1609
6.316.2 Constructor & Destructor Documentation	1609
6.316.2.1 ~Delayed	1609
6.316.3 Member Function Documentation	1609
6.316.3.1 getDelay	1609
6.317cms::DeliveryMode Class Reference	1609
6.317.1 Detailed Description	1610
6.317.2 Member Enumeration Documentation	1610
6.317.2.1 DELIVERY_MODE	1610
6.317.3 Constructor & Destructor Documentation	1610
6.317.3.1 ~DeliveryMode	1610
6.318cms::Destination Class Reference	1610
6.318.1 Detailed Description	1611
6.318.2 Member Enumeration Documentation	1611
6.318.2.1 DestinationType	1611

6.318.3 Constructor & Destructor Documentation	1612
6.318.3.1 ~Destination	1612
6.318.4 Member Function Documentation	1612
6.318.4.1 clone	1612
6.318.4.2 copy	1612
6.318.4.3 getCMSProperties	1612
6.318.4.4 getDestinationType	1612
6.319activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference . .	1613
6.319.1 Field Documentation	1613
6.319.1.1 ANY_CHILD	1613
6.319.1.2 ANY_DESCENDENT	1613
6.320activemq::commands::DestinationInfo Class Reference	1613
6.320.1 Constructor & Destructor Documentation	1615
6.320.1.1 DestinationInfo	1615
6.320.1.2 ~DestinationInfo	1615
6.320.2 Member Function Documentation	1615
6.320.2.1 cloneDataStructure	1615
6.320.2.2 copyDataStructure	1615
6.320.2.3 equals	1615
6.320.2.4 getBrokerPath	1616
6.320.2.5 getBrokerPath	1616
6.320.2.6 getConnectionId	1616
6.320.2.7 getConnectionId	1616
6.320.2.8 getDataStructureType	1616
6.320.2.9 getDestination	1617
6.320.2.10getDestination	1617
6.320.2.11getOperationType	1617
6.320.2.12getTimeout	1617
6.320.2.13setBrokerPath	1617
6.320.2.14setConnectionId	1617
6.320.2.15setDestination	1617
6.320.2.16setOperationType	1617
6.320.2.17setTimeout	1617
6.320.2.18oString	1617
6.320.2.19visit	1617
6.320.3 Field Documentation	1618

6.320.3.1 brokerPath	1618
6.320.3.2 connectionId	1618
6.320.3.3 destination	1618
6.320.3.4 ID_DESTINATIONINFO	1618
6.320.3.5 operationType	1618
6.320.3.6 timeout	1618
6.321activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller Class Reference	1618
6.321.1 Detailed Description	1619
6.321.2 Constructor & Destructor Documentation	1619
6.321.2.1 DestinationInfoMarshaller	1619
6.321.2.2 ~DestinationInfoMarshaller	1619
6.321.3 Member Function Documentation	1619
6.321.3.1 createObject	1619
6.321.3.2 getDataStructureType	1620
6.321.3.3 looseMarshal	1620
6.321.3.4 looseUnmarshal	1620
6.321.3.5 tightMarshal1	1621
6.321.3.6 tightMarshal2	1621
6.321.3.7 tightUnmarshal	1621
6.322activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller Class Reference	1622
6.322.1 Detailed Description	1623
6.322.2 Constructor & Destructor Documentation	1623
6.322.2.1 DestinationInfoMarshaller	1623
6.322.2.2 ~DestinationInfoMarshaller	1623
6.322.3 Member Function Documentation	1623
6.322.3.1 createObject	1623
6.322.3.2 getDataStructureType	1623
6.322.3.3 looseMarshal	1624
6.322.3.4 looseUnmarshal	1624
6.322.3.5 tightMarshal1	1624
6.322.3.6 tightMarshal2	1625
6.322.3.7 tightUnmarshal	1625
6.323activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller Class Reference	1626
6.323.1 Detailed Description	1627

6.323.2 Constructor & Destructor Documentation	1627
6.323.2.1 DestinationInfoMarshaller	1627
6.323.2.2 ~DestinationInfoMarshaller	1627
6.323.3 Member Function Documentation	1627
6.323.3.1 createObject	1627
6.323.3.2 getDataStructureType	1627
6.323.3.3 looseMarshal	1628
6.323.3.4 looseUnmarshal	1628
6.323.3.5 tightMarshal1	1628
6.323.3.6 tightMarshal2	1629
6.323.3.7 tightUnmarshal	1629
6.324activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller Class Reference	1630
6.324.1 Detailed Description	1631
6.324.2 Constructor & Destructor Documentation	1631
6.324.2.1 DestinationInfoMarshaller	1631
6.324.2.2 ~DestinationInfoMarshaller	1631
6.324.3 Member Function Documentation	1631
6.324.3.1 createObject	1631
6.324.3.2 getDataStructureType	1631
6.324.3.3 looseMarshal	1632
6.324.3.4 looseUnmarshal	1632
6.324.3.5 tightMarshal1	1632
6.324.3.6 tightMarshal2	1633
6.324.3.7 tightUnmarshal	1633
6.325activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller Class Reference	1634
6.325.1 Detailed Description	1635
6.325.2 Constructor & Destructor Documentation	1635
6.325.2.1 DestinationInfoMarshaller	1635
6.325.2.2 ~DestinationInfoMarshaller	1635
6.325.3 Member Function Documentation	1635
6.325.3.1 createObject	1635
6.325.3.2 getDataStructureType	1635
6.325.3.3 looseMarshal	1636
6.325.3.4 looseUnmarshal	1636
6.325.3.5 tightMarshal1	1636

6.325.3.6 tightMarshal2	1637
6.325.3.7 tightUnmarshal	1637
6.326activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller Class Reference	1638
6.326.1 Detailed Description	1639
6.326.2 Constructor & Destructor Documentation	1639
6.326.2.1 DestinationInfoMarshaller	1639
6.326.2.2 ~DestinationInfoMarshaller	1639
6.326.3 Member Function Documentation	1639
6.326.3.1 createObject	1639
6.326.3.2 getDataStructureType	1639
6.326.3.3 looseMarshal	1640
6.326.3.4 looseUnmarshal	1640
6.326.3.5 tightMarshal1	1640
6.326.3.6 tightMarshal2	1641
6.326.3.7 tightUnmarshal	1641
6.327activemq::cmsutil::DestinationResolver Class Reference	1642
6.327.1 Detailed Description	1642
6.327.2 Constructor & Destructor Documentation	1642
6.327.2.1 ~DestinationResolver	1642
6.327.3 Member Function Documentation	1642
6.327.3.1 destroy	1642
6.327.3.2 init	1643
6.327.3.3 resolveDestinationName	1643
6.328activemq::commands::DiscoveryEvent Class Reference	1643
6.328.1 Constructor & Destructor Documentation	1645
6.328.1.1 DiscoveryEvent	1645
6.328.1.2 ~DiscoveryEvent	1645
6.328.2 Member Function Documentation	1645
6.328.2.1 cloneDataStructure	1645
6.328.2.2 copyDataStructure	1645
6.328.2.3 equals	1645
6.328.2.4 getBrokerName	1646
6.328.2.5 getBrokerName	1646
6.328.2.6 getDataStructureType	1646
6.328.2.7 getServiceName	1646
6.328.2.8 getServiceName	1646

6.328.2.9	setBrokerName	1646
6.328.2.10	getServiceName	1646
6.328.2.11	toString	1646
6.328.3	Field Documentation	1647
6.328.3.1	brokerName	1647
6.328.3.2	ID_DISCOVERYEVENT	1647
6.328.3.3	serviceName	1647
6.329	activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller Class Reference	1647
6.329.1	Detailed Description	1648
6.329.2	Constructor & Destructor Documentation	1648
6.329.2.1	DiscoveryEventMarshaller	1648
6.329.2.2	~DiscoveryEventMarshaller	1648
6.329.3	Member Function Documentation	1648
6.329.3.1	createObject	1648
6.329.3.2	getDataStructureType	1648
6.329.3.3	looseMarshal	1649
6.329.3.4	looseUnmarshal	1649
6.329.3.5	tightMarshal1	1649
6.329.3.6	tightMarshal2	1650
6.329.3.7	tightUnmarshal	1650
6.330	activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller Class Reference	1651
6.330.1	Detailed Description	1652
6.330.2	Constructor & Destructor Documentation	1652
6.330.2.1	DiscoveryEventMarshaller	1652
6.330.2.2	~DiscoveryEventMarshaller	1652
6.330.3	Member Function Documentation	1652
6.330.3.1	createObject	1652
6.330.3.2	getDataStructureType	1652
6.330.3.3	looseMarshal	1652
6.330.3.4	looseUnmarshal	1653
6.330.3.5	tightMarshal1	1653
6.330.3.6	tightMarshal2	1654
6.330.3.7	tightUnmarshal	1654
6.331	activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller Class Reference	1654

6.331.1 Detailed Description	1655
6.331.2 Constructor & Destructor Documentation	1656
6.331.2.1 DiscoveryEventMarshaller	1656
6.331.2.2 ~DiscoveryEventMarshaller	1656
6.331.3 Member Function Documentation	1656
6.331.3.1 createObject	1656
6.331.3.2 getDataStructureType	1656
6.331.3.3 looseMarshal	1656
6.331.3.4 looseUnmarshal	1657
6.331.3.5 tightMarshal1	1657
6.331.3.6 tightMarshal2	1657
6.331.3.7 tightUnmarshal	1658
6.332activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller Class Reference	1658
6.332.1 Detailed Description	1659
6.332.2 Constructor & Destructor Documentation	1660
6.332.2.1 DiscoveryEventMarshaller	1660
6.332.2.2 ~DiscoveryEventMarshaller	1660
6.332.3 Member Function Documentation	1660
6.332.3.1 createObject	1660
6.332.3.2 getDataStructureType	1660
6.332.3.3 looseMarshal	1660
6.332.3.4 looseUnmarshal	1661
6.332.3.5 tightMarshal1	1661
6.332.3.6 tightMarshal2	1661
6.332.3.7 tightUnmarshal	1662
6.333activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller Class Reference	1662
6.333.1 Detailed Description	1663
6.333.2 Constructor & Destructor Documentation	1664
6.333.2.1 DiscoveryEventMarshaller	1664
6.333.2.2 ~DiscoveryEventMarshaller	1664
6.333.3 Member Function Documentation	1664
6.333.3.1 createObject	1664
6.333.3.2 getDataStructureType	1664
6.333.3.3 looseMarshal	1664
6.333.3.4 looseUnmarshal	1665

6.333.3.5 tightMarshal1	1665
6.333.3.6 tightMarshal2	1665
6.333.3.7 tightUnmarshal	1666
6.334activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller Class Reference	1666
6.334.1 Detailed Description	1667
6.334.2 Constructor & Destructor Documentation	1668
6.334.2.1 DiscoveryEventMarshaller	1668
6.334.2.2 ~DiscoveryEventMarshaller	1668
6.334.3 Member Function Documentation	1668
6.334.3.1 createObject	1668
6.334.3.2 getDataStructureType	1668
6.334.3.3 looseMarshal	1668
6.334.3.4 looseUnmarshal	1669
6.334.3.5 tightMarshal1	1669
6.334.3.6 tightMarshal2	1669
6.334.3.7 tightUnmarshal	1670
6.335activemq::core::DispatchData Class Reference	1670
6.335.1 Detailed Description	1671
6.335.2 Constructor & Destructor Documentation	1671
6.335.2.1 DispatchData	1671
6.335.2.2 DispatchData	1671
6.335.3 Member Function Documentation	1671
6.335.3.1 getConsumerId	1671
6.335.3.2 getMessage	1671
6.336activemq::core::Dispatcher Class Reference	1671
6.336.1 Detailed Description	1671
6.336.2 Constructor & Destructor Documentation	1672
6.336.2.1 ~Dispatcher	1672
6.336.3 Member Function Documentation	1672
6.336.3.1 dispatch	1672
6.337decaf::lang::Double Class Reference	1672
6.337.1 Constructor & Destructor Documentation	1674
6.337.1.1 Double	1674
6.337.1.2 Double	1674
6.337.1.3 ~Double	1675
6.337.2 Member Function Documentation	1675

6.337.2.1 byteValue	1675
6.337.2.2 compare	1675
6.337.2.3 compareTo	1675
6.337.2.4 compareTo	1675
6.337.2.5 doubleToLongBits	1676
6.337.2.6 doubleToRawLongBits	1676
6.337.2.7 doubleValue	1677
6.337.2.8 equals	1677
6.337.2.9 equals	1677
6.337.2.10 float Value	1677
6.337.2.11 int Value	1678
6.337.2.12 isInfinite	1678
6.337.2.13 isInfinite	1678
6.337.2.14 isNaN	1678
6.337.2.15 isNaN	1678
6.337.2.16 longBitsToDouble	1678
6.337.2.17 long Value	1679
6.337.2.18 operator<	1679
6.337.2.19 operator<	1679
6.337.2.20 operator==	1680
6.337.2.21 operator==	1680
6.337.2.22 parseDouble	1680
6.337.2.23 short Value	1680
6.337.2.24 toHexString	1681
6.337.2.25 toString	1681
6.337.2.26 toString	1681
6.337.2.27 valueOf	1682
6.337.2.28 valueOf	1682
6.337.3 Field Documentation	1682
6.337.3.1 MAX_VALUE	1682
6.337.3.2 MIN_VALUE	1682
6.337.3.3 NaN	1683
6.337.3.4 NEGATIVE_INFINITY	1683
6.337.3.5 POSITIVE_INFINITY	1683
6.337.3.6 SIZE	1683
6.338 decaf::internal::nio::DoubleArrayBuffer Class Reference	1683

6.338.1 Constructor & Destructor Documentation	1686
6.338.1.1 DoubleArrayBuffer	1686
6.338.1.2 DoubleArrayBuffer	1687
6.338.1.3 DoubleArrayBuffer	1687
6.338.1.4 DoubleArrayBuffer	1688
6.338.1.5 ~DoubleArrayBuffer	1688
6.338.2 Member Function Documentation	1688
6.338.2.1 array	1688
6.338.2.2 arrayOffset	1688
6.338.2.3 asReadOnlyBuffer	1689
6.338.2.4 compact	1689
6.338.2.5 duplicate	1689
6.338.2.6 get	1690
6.338.2.7 get	1690
6.338.2.8 hasArray	1691
6.338.2.9 isReadOnly	1691
6.338.2.10put	1691
6.338.2.11put	1691
6.338.2.12setReadOnly	1692
6.338.2.13lice	1692
6.339decaf::nio::DoubleBuffer Class Reference	1692
6.339.1 Detailed Description	1695
6.339.2 Constructor & Destructor Documentation	1695
6.339.2.1 DoubleBuffer	1695
6.339.2.2 ~DoubleBuffer	1695
6.339.3 Member Function Documentation	1695
6.339.3.1 allocate	1695
6.339.3.2 array	1696
6.339.3.3 arrayOffset	1696
6.339.3.4 asReadOnlyBuffer	1696
6.339.3.5 compact	1697
6.339.3.6 compareTo	1697
6.339.3.7 duplicate	1697
6.339.3.8 equals	1698
6.339.3.9 get	1698
6.339.3.10get	1698

6.339.3.11	get	1698
6.339.3.12	get	1699
6.339.3.13	hasArray	1699
6.339.3.14	operator<	1700
6.339.3.15	operator==	1700
6.339.3.16	put	1700
6.339.3.17	put	1700
6.339.3.18	put	1701
6.339.3.19	put	1701
6.339.3.20	put	1702
6.339.3.21	slice	1702
6.339.3.22	toString	1702
6.339.3.23	wrap	1703
6.339.3.24	wrap	1703
6.340	decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference	1704
6.341	activemq::cmsutil::DynamicDestinationResolver Class Reference	1704
6.341.1	Detailed Description	1704
6.341.2	Constructor & Destructor Documentation	1705
6.341.2.1	DynamicDestinationResolver	1705
6.341.2.2	DynamicDestinationResolver	1705
6.341.2.3	~DynamicDestinationResolver	1705
6.341.3	Member Function Documentation	1705
6.341.3.1	destroy	1705
6.341.3.2	init	1705
6.341.3.3	operator=	1705
6.341.3.4	resolveDestinationName	1705
6.342	decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference	1706
6.342.1	Constructor & Destructor Documentation	1707
6.342.1.1	Entry	1707
6.342.1.2	~Entry	1707
6.342.2	Member Function Documentation	1707
6.342.2.1	getKey	1707
6.342.2.2	getValue	1707
6.342.2.3	setValue	1707
6.343	decaf::io::EOFException Class Reference	1707
6.343.1	Constructor & Destructor Documentation	1708

6.343.1.1 EOFException	1708
6.343.1.2 EOFException	1708
6.343.1.3 EOFException	1708
6.343.1.4 EOFException	1708
6.343.1.5 EOFException	1709
6.343.1.6 EOFException	1709
6.343.1.7 ~EOFException	1709
6.343.2 Member Function Documentation	1709
6.343.2.1 clone	1709
6.344decaf::util::logging::ErrorManager Class Reference	1710
6.344.1 Detailed Description	1710
6.344.2 Constructor & Destructor Documentation	1711
6.344.2.1 ErrorManager	1711
6.344.2.2 ~ErrorManager	1711
6.344.3 Member Function Documentation	1711
6.344.3.1 error	1711
6.344.4 Field Documentation	1711
6.344.4.1 CLOSE_FAILURE	1711
6.344.4.2 FLUSH_FAILURE	1711
6.344.4.3 FORMAT_FAILURE	1711
6.344.4.4 GENERIC_FAILURE	1711
6.344.4.5 OPEN_FAILURE	1711
6.344.4.6 WRITE_FAILURE	1712
6.345decaf::lang::Exception Class Reference	1712
6.345.1 Constructor & Destructor Documentation	1713
6.345.1.1 Exception	1713
6.345.1.2 Exception	1714
6.345.1.3 Exception	1714
6.345.1.4 Exception	1714
6.345.1.5 Exception	1714
6.345.1.6 ~Exception	1715
6.345.2 Member Function Documentation	1715
6.345.2.1 buildMessage	1715
6.345.2.2 clone	1715
6.345.2.3 getCause	1716
6.345.2.4 getMessage	1716

6.345.2.5	getStackTrace	1716
6.345.2.6	getStackTraceString	1717
6.345.2.7	initCause	1717
6.345.2.8	operator=	1717
6.345.2.9	printStackTrace	1717
6.345.2.10	printStackTrace	1717
6.345.2.11	setMark	1718
6.345.2.12	setMessage	1718
6.345.2.13	setStackTrace	1718
6.345.2.14	what	1718
6.345.3	Field Documentation	1718
6.345.3.1	cause	1718
6.345.3.2	message	1718
6.345.3.3	stackTrace	1719
6.346	cms::ExceptionListener Class Reference	1719
6.346.1	Detailed Description	1719
6.346.2	Constructor & Destructor Documentation	1719
6.346.2.1	~ExceptionListener	1719
6.346.3	Member Function Documentation	1719
6.346.3.1	onException	1719
6.347	activemq::commands::ExceptionResponse Class Reference	1720
6.347.1	Constructor & Destructor Documentation	1721
6.347.1.1	ExceptionResponse	1721
6.347.1.2	~ExceptionResponse	1721
6.347.2	Member Function Documentation	1721
6.347.2.1	cloneDataStructure	1721
6.347.2.2	copyDataStructure	1721
6.347.2.3	equals	1721
6.347.2.4	getDataStructureType	1721
6.347.2.5	getException	1722
6.347.2.6	getException	1722
6.347.2.7	setException	1722
6.347.2.8	toString	1722
6.347.3	Field Documentation	1722
6.347.3.1	exception	1722
6.347.3.2	ID_EXCEPTIONRESPONSE	1722

6.348	activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller	Class	
	Reference		1722
6.348.1	Detailed Description		1723
6.348.2	Constructor & Destructor Documentation		1724
	6.348.2.1 ExceptionResponseMarshaller		1724
	6.348.2.2 ~ExceptionResponseMarshaller		1724
6.348.3	Member Function Documentation		1724
	6.348.3.1 createObject		1724
	6.348.3.2 getDataStructureType		1724
	6.348.3.3 looseMarshal		1724
	6.348.3.4 looseUnmarshal		1725
	6.348.3.5 tightMarshal1		1725
	6.348.3.6 tightMarshal2		1726
	6.348.3.7 tightUnmarshal		1726
6.349	activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	Class	
	Reference		1726
6.349.1	Detailed Description		1727
6.349.2	Constructor & Destructor Documentation		1728
	6.349.2.1 ExceptionResponseMarshaller		1728
	6.349.2.2 ~ExceptionResponseMarshaller		1728
6.349.3	Member Function Documentation		1728
	6.349.3.1 createObject		1728
	6.349.3.2 getDataStructureType		1728
	6.349.3.3 looseMarshal		1728
	6.349.3.4 looseUnmarshal		1729
	6.349.3.5 tightMarshal1		1729
	6.349.3.6 tightMarshal2		1730
	6.349.3.7 tightUnmarshal		1730
6.350	activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	Class	
	Reference		1730
6.350.1	Detailed Description		1731
6.350.2	Constructor & Destructor Documentation		1732
	6.350.2.1 ExceptionResponseMarshaller		1732
	6.350.2.2 ~ExceptionResponseMarshaller		1732
6.350.3	Member Function Documentation		1732
	6.350.3.1 createObject		1732
	6.350.3.2 getDataStructureType		1732

6.350.3.3 looseMarshal	1732
6.350.3.4 looseUnmarshal	1733
6.350.3.5 tightMarshal1	1733
6.350.3.6 tightMarshal2	1734
6.350.3.7 tightUnmarshal	1734
6.351activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller Class	
Reference	1734
6.351.1 Detailed Description	1735
6.351.2 Constructor & Destructor Documentation	1736
6.351.2.1 ExceptionResponseMarshaller	1736
6.351.2.2 ~ExceptionResponseMarshaller	1736
6.351.3 Member Function Documentation	1736
6.351.3.1 createObject	1736
6.351.3.2 getDataStructureType	1736
6.351.3.3 looseMarshal	1736
6.351.3.4 looseUnmarshal	1737
6.351.3.5 tightMarshal1	1737
6.351.3.6 tightMarshal2	1738
6.351.3.7 tightUnmarshal	1738
6.352activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller Class	
Reference	1738
6.352.1 Detailed Description	1739
6.352.2 Constructor & Destructor Documentation	1740
6.352.2.1 ExceptionResponseMarshaller	1740
6.352.2.2 ~ExceptionResponseMarshaller	1740
6.352.3 Member Function Documentation	1740
6.352.3.1 createObject	1740
6.352.3.2 getDataStructureType	1740
6.352.3.3 looseMarshal	1740
6.352.3.4 looseUnmarshal	1741
6.352.3.5 tightMarshal1	1741
6.352.3.6 tightMarshal2	1742
6.352.3.7 tightUnmarshal	1742
6.353activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller Class	
Reference	1742
6.353.1 Detailed Description	1743
6.353.2 Constructor & Destructor Documentation	1744

6.353.2.1	ExceptionResponseMarshaller	1744
6.353.2.2	~ExceptionResponseMarshaller	1744
6.353.3	Member Function Documentation	1744
6.353.3.1	createObject	1744
6.353.3.2	getDataStructureType	1744
6.353.3.3	looseMarshal	1744
6.353.3.4	looseUnmarshal	1745
6.353.3.5	tightMarshal1	1745
6.353.3.6	tightMarshal2	1746
6.353.3.7	tightUnmarshal	1746
6.354	decaf::util::concurrent::ExecutionException Class Reference	1746
6.354.1	Constructor & Destructor Documentation	1747
6.354.1.1	ExecutionException	1747
6.354.1.2	ExecutionException	1747
6.354.1.3	ExecutionException	1748
6.354.1.4	ExecutionException	1748
6.354.1.5	ExecutionException	1748
6.354.1.6	ExecutionException	1748
6.354.1.7	~ExecutionException	1749
6.354.2	Member Function Documentation	1749
6.354.2.1	clone	1749
6.355	decaf::util::concurrent::Executor Class Reference	1749
6.355.1	Detailed Description	1749
6.355.2	Constructor & Destructor Documentation	1750
6.355.2.1	~Executor	1750
6.355.3	Member Function Documentation	1750
6.355.3.1	execute	1750
6.356	decaf::util::concurrent::ExecutorService Class Reference	1751
6.356.1	Detailed Description	1751
6.356.2	Constructor & Destructor Documentation	1752
6.356.2.1	~ExecutorService	1752
6.356.3	Member Function Documentation	1752
6.356.3.1	awaitTermination	1752
6.357	activemq::transport::failover::FailoverTransport Class Reference	1752
6.357.1	Constructor & Destructor Documentation	1755
6.357.1.1	FailoverTransport	1755

6.357.1.2 ~FailoverTransport	1755
6.357.2 Member Function Documentation	1755
6.357.2.1 add	1755
6.357.2.2 addURI	1755
6.357.2.3 close	1755
6.357.2.4 getBackOffMultiplier	1756
6.357.2.5 getBackupPoolSize	1756
6.357.2.6 getInitialReconnectDelay	1756
6.357.2.7 getMaxCacheSize	1756
6.357.2.8 getMaxReconnectAttempts	1756
6.357.2.9 getMaxReconnectDelay	1756
6.357.2.10 getReconnectDelay	1756
6.357.2.11 getRemoteAddress	1756
6.357.2.12 getStartupMaxReconnectAttempts	1756
6.357.2.13 getTimeout	1756
6.357.2.14 getTransportListener	1756
6.357.2.15 handleTransportFailure	1757
6.357.2.16 isBackup	1757
6.357.2.17 isClosed	1757
6.357.2.18 isConnected	1757
6.357.2.19 isFaultTolerant	1757
6.357.2.20 isInitialized	1758
6.357.2.21 isPending	1758
6.357.2.22 isRandomize	1758
6.357.2.23 isTrackMessages	1758
6.357.2.24 isTrackTransactionProducers	1758
6.357.2.25 isUseExponentialBackOff	1758
6.357.2.26 iterate	1758
6.357.2.27 narrow	1758
6.357.2.28 neway	1759
6.357.2.29 reconnect	1759
6.357.2.30 reconnect	1759
6.357.2.31 removeURI	1759
6.357.2.32 request	1760
6.357.2.33 request	1760
6.357.2.34 restoreTransport	1760

6.357.2.35	setBackOffMultiplier	1761
6.357.2.36	setBackup	1761
6.357.2.37	setBackupPoolSize	1761
6.357.2.38	setConnectionInterruptProcessingComplete	1761
6.357.2.39	setInitialized	1761
6.357.2.40	setInitialReconnectDelay	1762
6.357.2.41	setMaxCacheSize	1762
6.357.2.42	setMaxReconnectAttempts	1762
6.357.2.43	setMaxReconnectDelay	1762
6.357.2.44	setRandomize	1762
6.357.2.45	setReconnectDelay	1762
6.357.2.46	setStartupMaxReconnectAttempts	1762
6.357.2.47	setTimeout	1762
6.357.2.48	setTrackMessages	1762
6.357.2.49	setTrackTransactionProducers	1762
6.357.2.50	setTransportListener	1762
6.357.2.51	setUseExponentialBackOff	1763
6.357.2.52	setWireFormat	1763
6.357.2.53	start	1763
6.357.2.54	stop	1763
6.357.3	Friends And Related Function Documentation	1763
6.357.3.1	FailoverTransportListener	1763
6.358	activemq::transport::failover::FailoverTransportFactory Class Reference	1763
6.358.1	Detailed Description	1764
6.358.2	Constructor & Destructor Documentation	1764
6.358.2.1	~FailoverTransportFactory	1764
6.358.3	Member Function Documentation	1764
6.358.3.1	create	1764
6.358.3.2	createComposite	1765
6.358.3.3	doCreateComposite	1765
6.359	activemq::transport::failover::FailoverTransportListener Class Reference	1766
6.359.1	Detailed Description	1766
6.359.2	Constructor & Destructor Documentation	1767
6.359.2.1	FailoverTransportListener	1767
6.359.2.2	~FailoverTransportListener	1767
6.359.3	Member Function Documentation	1767

6.359.3.1 onCommand	1767
6.359.3.2 onException	1767
6.359.3.3 transportInterrupted	1767
6.359.3.4 transportResumed	1767
6.360decaf::io::FileDescriptor Class Reference	1768
6.360.1 Detailed Description	1769
6.360.2 Constructor & Destructor Documentation	1769
6.360.2.1 FileDescriptor	1769
6.360.2.2 FileDescriptor	1769
6.360.2.3 ~FileDescriptor	1769
6.360.3 Member Function Documentation	1769
6.360.3.1 sync	1769
6.360.3.2 valid	1769
6.360.4 Field Documentation	1769
6.360.4.1 descriptor	1769
6.360.4.2 err	1769
6.360.4.3 in	1769
6.360.4.4 out	1770
6.360.4.5 readonly	1770
6.361decaf::util::logging::Filter Class Reference	1770
6.361.1 Detailed Description	1770
6.361.2 Constructor & Destructor Documentation	1770
6.361.2.1 ~Filter	1770
6.361.3 Member Function Documentation	1770
6.361.3.1 isLoggable	1770
6.362decaf::io::FilterInputStream Class Reference	1771
6.362.1 Detailed Description	1773
6.362.2 Constructor & Destructor Documentation	1773
6.362.2.1 FilterInputStream	1773
6.362.2.2 ~FilterInputStream	1773
6.362.3 Member Function Documentation	1773
6.362.3.1 available	1773
6.362.3.2 close	1774
6.362.3.3 doReadArray	1774
6.362.3.4 doReadArrayBounded	1774
6.362.3.5 doReadByte	1774

6.362.3.6	isClosed	1775
6.362.3.7	mark	1775
6.362.3.8	markSupported	1775
6.362.3.9	reset	1775
6.362.3.10	skip	1776
6.362.4	Field Documentation	1777
6.362.4.1	closed	1777
6.362.4.2	inputStream	1777
6.362.4.3	own	1777
6.363	decaf::io::FilterOutputStream Class Reference	1777
6.363.1	Detailed Description	1778
6.363.2	Constructor & Destructor Documentation	1778
6.363.2.1	FilterOutputStream	1778
6.363.2.2	~FilterOutputStream	1778
6.363.3	Member Function Documentation	1778
6.363.3.1	close	1778
6.363.3.2	doWriteArray	1779
6.363.3.3	doWriteArrayBounded	1779
6.363.3.4	doWriteByte	1779
6.363.3.5	flush	1779
6.363.3.6	isClosed	1779
6.363.3.7	toString	1780
6.363.4	Field Documentation	1780
6.363.4.1	closed	1780
6.363.4.2	outputStream	1780
6.363.4.3	own	1780
6.364	decaf::lang::Float Class Reference	1780
6.364.1	Constructor & Destructor Documentation	1782
6.364.1.1	Float	1782
6.364.1.2	Float	1783
6.364.1.3	Float	1783
6.364.1.4	~Float	1783
6.364.2	Member Function Documentation	1783
6.364.2.1	byteValue	1783
6.364.2.2	compare	1783
6.364.2.3	compareTo	1783

6.364.2.4	compareTo	1784
6.364.2.5	doubleValue	1784
6.364.2.6	equals	1784
6.364.2.7	equals	1784
6.364.2.8	floatToIntBits	1785
6.364.2.9	floatToRawIntBits	1785
6.364.2.10	floatValue	1785
6.364.2.11	intBitsToFloat	1786
6.364.2.12	intValue	1786
6.364.2.13	isInfinite	1786
6.364.2.14	isInfinite	1786
6.364.2.15	isNaN	1786
6.364.2.16	isNaN	1787
6.364.2.17	longValue	1787
6.364.2.18	operator<	1787
6.364.2.19	operator<	1787
6.364.2.20	operator==	1788
6.364.2.21	operator==	1788
6.364.2.22	parseFloat	1788
6.364.2.23	shortValue	1788
6.364.2.24	toHexString	1789
6.364.2.25	toString	1789
6.364.2.26	toString	1789
6.364.2.27	valueOf	1790
6.364.2.28	valueOf	1790
6.364.3	Field Documentation	1790
6.364.3.1	MAX_VALUE	1790
6.364.3.2	MIN_VALUE	1790
6.364.3.3	NaN	1791
6.364.3.4	NEGATIVE_INFINITY	1791
6.364.3.5	POSITIVE_INFINITY	1791
6.364.3.6	SIZE	1791
6.365	decaf::internal::nio::FloatArrayBuffer Class Reference	1791
6.365.1	Constructor & Destructor Documentation	1794
6.365.1.1	FloatArrayBuffer	1794
6.365.1.2	FloatArrayBuffer	1795

6.365.1.3 FloatArrayBuffer	1795
6.365.1.4 FloatArrayBuffer	1795
6.365.1.5 ~FloatArrayBuffer	1796
6.365.2 Member Function Documentation	1796
6.365.2.1 array	1796
6.365.2.2 arrayOffset	1796
6.365.2.3 asReadOnlyBuffer	1796
6.365.2.4 compact	1797
6.365.2.5 duplicate	1797
6.365.2.6 get	1798
6.365.2.7 get	1798
6.365.2.8 hasArray	1798
6.365.2.9 isReadOnly	1799
6.365.2.10put	1799
6.365.2.11put	1799
6.365.2.12setReadOnly	1799
6.365.2.13slice	1800
6.366decaf::nio::FloatBuffer Class Reference	1800
6.366.1 Detailed Description	1802
6.366.2 Constructor & Destructor Documentation	1802
6.366.2.1 FloatBuffer	1802
6.366.2.2 ~FloatBuffer	1803
6.366.3 Member Function Documentation	1803
6.366.3.1 allocate	1803
6.366.3.2 array	1803
6.366.3.3 arrayOffset	1804
6.366.3.4 asReadOnlyBuffer	1804
6.366.3.5 compact	1804
6.366.3.6 compareTo	1805
6.366.3.7 duplicate	1805
6.366.3.8 equals	1805
6.366.3.9 get	1805
6.366.3.10get	1806
6.366.3.11get	1806
6.366.3.12get	1807
6.366.3.13hasArray	1807

6.366.3.14	operator<	1807
6.366.3.15	operator==	1807
6.366.3.16	put	1807
6.366.3.17	put	1808
6.366.3.18	put	1808
6.366.3.19	put	1809
6.366.3.20	put	1809
6.366.3.21	slice	1809
6.366.3.22	toString	1810
6.366.3.23	wrap	1810
6.366.3.24	wrap	1810
6.367	decaf::io::Flushable Class Reference	1811
6.367.1	Detailed Description	1811
6.367.2	Constructor & Destructor Documentation	1811
6.367.2.1	~Flushable	1811
6.367.3	Member Function Documentation	1811
6.367.3.1	flush	1811
6.368	activemq::commands::FlushCommand Class Reference	1812
6.368.1	Constructor & Destructor Documentation	1813
6.368.1.1	FlushCommand	1813
6.368.1.2	~FlushCommand	1813
6.368.2	Member Function Documentation	1813
6.368.2.1	cloneDataStructure	1813
6.368.2.2	copyDataStructure	1813
6.368.2.3	equals	1813
6.368.2.4	getDataStructureType	1813
6.368.2.5	toString	1814
6.368.2.6	visit	1814
6.368.3	Field Documentation	1814
6.368.3.1	ID_FLUSHCOMMAND	1814
6.369	activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller Class Reference	1814
6.369.1	Detailed Description	1815
6.369.2	Constructor & Destructor Documentation	1816
6.369.2.1	FlushCommandMarshaller	1816
6.369.2.2	~FlushCommandMarshaller	1816
6.369.3	Member Function Documentation	1816

6.369.3.1 createObject	1816
6.369.3.2 getDataStructureType	1816
6.369.3.3 looseMarshal	1816
6.369.3.4 looseUnmarshal	1817
6.369.3.5 tightMarshal1	1817
6.369.3.6 tightMarshal2	1817
6.369.3.7 tightUnmarshal	1818
6.370activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller Class Reference	1818
6.370.1 Detailed Description	1819
6.370.2 Constructor & Destructor Documentation	1820
6.370.2.1 FlushCommandMarshaller	1820
6.370.2.2 ~FlushCommandMarshaller	1820
6.370.3 Member Function Documentation	1820
6.370.3.1 createObject	1820
6.370.3.2 getDataStructureType	1820
6.370.3.3 looseMarshal	1820
6.370.3.4 looseUnmarshal	1821
6.370.3.5 tightMarshal1	1821
6.370.3.6 tightMarshal2	1821
6.370.3.7 tightUnmarshal	1822
6.371activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller Class Reference	1822
6.371.1 Detailed Description	1823
6.371.2 Constructor & Destructor Documentation	1824
6.371.2.1 FlushCommandMarshaller	1824
6.371.2.2 ~FlushCommandMarshaller	1824
6.371.3 Member Function Documentation	1824
6.371.3.1 createObject	1824
6.371.3.2 getDataStructureType	1824
6.371.3.3 looseMarshal	1824
6.371.3.4 looseUnmarshal	1825
6.371.3.5 tightMarshal1	1825
6.371.3.6 tightMarshal2	1825
6.371.3.7 tightUnmarshal	1826
6.372activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller Class Reference	1826

6.372.1 Detailed Description	1827
6.372.2 Constructor & Destructor Documentation	1828
6.372.2.1 FlushCommandMarshaller	1828
6.372.2.2 ~FlushCommandMarshaller	1828
6.372.3 Member Function Documentation	1828
6.372.3.1 createObject	1828
6.372.3.2 getDataStructureType	1828
6.372.3.3 looseMarshal	1828
6.372.3.4 looseUnmarshal	1829
6.372.3.5 tightMarshal1	1829
6.372.3.6 tightMarshal2	1829
6.372.3.7 tightUnmarshal	1830
6.373activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller Class Reference	1830
6.373.1 Detailed Description	1831
6.373.2 Constructor & Destructor Documentation	1832
6.373.2.1 FlushCommandMarshaller	1832
6.373.2.2 ~FlushCommandMarshaller	1832
6.373.3 Member Function Documentation	1832
6.373.3.1 createObject	1832
6.373.3.2 getDataStructureType	1832
6.373.3.3 looseMarshal	1832
6.373.3.4 looseUnmarshal	1833
6.373.3.5 tightMarshal1	1833
6.373.3.6 tightMarshal2	1833
6.373.3.7 tightUnmarshal	1834
6.374activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller Class Reference	1834
6.374.1 Detailed Description	1835
6.374.2 Constructor & Destructor Documentation	1836
6.374.2.1 FlushCommandMarshaller	1836
6.374.2.2 ~FlushCommandMarshaller	1836
6.374.3 Member Function Documentation	1836
6.374.3.1 createObject	1836
6.374.3.2 getDataStructureType	1836
6.374.3.3 looseMarshal	1836
6.374.3.4 looseUnmarshal	1837

6.374.3.5 tightMarshal1	1837
6.374.3.6 tightMarshal2	1837
6.374.3.7 tightUnmarshal	1838
6.375decaf::util::logging::Formatter Class Reference	1838
6.375.1 Detailed Description	1839
6.375.2 Constructor & Destructor Documentation	1839
6.375.2.1 ~Formatter	1839
6.375.3 Member Function Documentation	1839
6.375.3.1 format	1839
6.375.3.2 formatMessage	1839
6.375.3.3 getHead	1840
6.375.3.4 getTail	1840
6.376decaf::util::concurrent::Future< V > Class Template Reference	1840
6.376.1 Detailed Description	1841
6.376.2 Constructor & Destructor Documentation	1841
6.376.2.1 ~Future	1841
6.376.3 Member Function Documentation	1841
6.376.3.1 cancel	1841
6.376.3.2 get	1842
6.376.3.3 get	1842
6.376.3.4 isCancelled	1843
6.376.3.5 isDone	1843
6.377activemq::transport::correlator::FutureResponse Class Reference	1843
6.377.1 Detailed Description	1843
6.377.2 Constructor & Destructor Documentation	1844
6.377.2.1 FutureResponse	1844
6.377.2.2 ~FutureResponse	1844
6.377.3 Member Function Documentation	1844
6.377.3.1 getResponse	1844
6.377.3.2 getResponse	1844
6.377.3.3 getResponse	1844
6.377.3.4 getResponse	1844
6.377.3.5 setResponse	1844
6.378decaf::security::GeneralSecurityException Class Reference	1845
6.378.1 Constructor & Destructor Documentation	1845
6.378.1.1 GeneralSecurityException	1845

6.378.1.2 GeneralSecurityException	1846
6.378.1.3 GeneralSecurityException	1846
6.378.1.4 GeneralSecurityException	1846
6.378.1.5 GeneralSecurityException	1846
6.378.1.6 GeneralSecurityException	1846
6.378.1.7 ~GeneralSecurityException	1847
6.378.2 Member Function Documentation	1847
6.378.2.1 clone	1847
6.379decaf::internal::util::GenericResource< T > Class Template Reference	1847
6.379.1 Detailed Description	1848
6.379.2 Constructor & Destructor Documentation	1848
6.379.2.1 GenericResource	1848
6.379.2.2 ~GenericResource	1848
6.379.3 Member Function Documentation	1848
6.379.3.1 getManaged	1848
6.379.3.2 setManaged	1848
6.380gz_header_s Struct Reference	1848
6.380.1 Field Documentation	1849
6.380.1.1 comm_max	1849
6.380.1.2 comment	1849
6.380.1.3 done	1849
6.380.1.4 extra	1849
6.380.1.5 extra_len	1849
6.380.1.6 extra_max	1849
6.380.1.7 hcrc	1849
6.380.1.8 name	1849
6.380.1.9 name_max	1849
6.380.1.10bs	1849
6.380.1.11text	1849
6.380.1.12ime	1849
6.380.1.13flags	1849
6.381gz_state Struct Reference	1849
6.381.1 Field Documentation	1851
6.381.1.1 direct	1851
6.381.1.2 eof	1851
6.381.1.3 err	1851

6.381.1.4 fd	1851
6.381.1.5 have	1851
6.381.1.6 how	1851
6.381.1.7 in	1851
6.381.1.8 level	1851
6.381.1.9 mode	1851
6.381.1.10msg	1851
6.381.1.11next	1851
6.381.1.12out	1851
6.381.1.13path	1851
6.381.1.14pos	1851
6.381.1.15raw	1851
6.381.1.16seek	1851
6.381.1.17size	1851
6.381.1.18skip	1851
6.381.1.19start	1851
6.381.1.20strategy	1851
6.381.1.21strm	1851
6.381.1.22want	1851
6.382decaf::util::logging::Handler Class Reference	1852
6.382.1 Detailed Description	1853
6.382.2 Constructor & Destructor Documentation	1853
6.382.2.1 Handler	1853
6.382.2.2 ~Handler	1853
6.382.3 Member Function Documentation	1853
6.382.3.1 flush	1853
6.382.3.2 getErrorManager	1853
6.382.3.3 getFilter	1853
6.382.3.4 getFormatter	1854
6.382.3.5 getLevel	1854
6.382.3.6 isLoggable	1854
6.382.3.7 publish	1854
6.382.3.8 reportError	1854
6.382.3.9 setErrorManager	1855
6.382.3.10setFilter	1855
6.382.3.11setFormatter	1855

6.382.3.12	setLevel	1855
6.383	decaf::internal::util::HexStringParser Class Reference	1856
6.383.1	Constructor & Destructor Documentation	1856
6.383.1.1	HexStringParser	1856
6.383.1.2	~HexStringParser	1856
6.383.2	Member Function Documentation	1856
6.383.2.1	parse	1856
6.383.2.2	parseDouble	1857
6.383.2.3	parseFloat	1857
6.384	activemq::wireformat::openwire::utils::HexTable Class Reference	1857
6.384.1	Detailed Description	1857
6.384.2	Constructor & Destructor Documentation	1858
6.384.2.1	HexTable	1858
6.384.2.2	~HexTable	1858
6.384.3	Member Function Documentation	1858
6.384.3.1	operator[]	1858
6.384.3.2	operator[]	1858
6.384.3.3	size	1858
6.385	decaf::net::HttpRetryException Class Reference	1858
6.385.1	Constructor & Destructor Documentation	1859
6.385.1.1	HttpRetryException	1859
6.385.1.2	HttpRetryException	1859
6.385.1.3	HttpRetryException	1859
6.385.1.4	HttpRetryException	1860
6.385.1.5	HttpRetryException	1860
6.385.1.6	HttpRetryException	1860
6.385.1.7	~HttpRetryException	1861
6.385.2	Member Function Documentation	1861
6.385.2.1	clone	1861
6.386	activemq::util::IdGenerator Class Reference	1861
6.386.1	Constructor & Destructor Documentation	1862
6.386.1.1	IdGenerator	1862
6.386.1.2	IdGenerator	1862
6.386.1.3	~IdGenerator	1862
6.386.2	Member Function Documentation	1862
6.386.2.1	compare	1862

6.386.2.2 generateId	1862
6.386.2.3 getHostname	1862
6.386.2.4 getSeedFromId	1862
6.386.2.5 getSequenceFromId	1863
6.387decaf::lang::exceptions::IllegalArgumentException Class Reference	1863
6.387.1 Constructor & Destructor Documentation	1864
6.387.1.1 IllegalArgumentException	1864
6.387.1.2 IllegalArgumentException	1864
6.387.1.3 IllegalArgumentException	1864
6.387.1.4 IllegalArgumentException	1864
6.387.1.5 IllegalArgumentException	1864
6.387.1.6 IllegalArgumentException	1865
6.387.1.7 ~IllegalArgumentException	1865
6.387.2 Member Function Documentation	1865
6.387.2.1 clone	1865
6.388decaf::lang::exceptions::IllegalMonitorStateException Class Reference	1865
6.388.1 Constructor & Destructor Documentation	1866
6.388.1.1 IllegalMonitorStateException	1866
6.388.1.2 IllegalMonitorStateException	1866
6.388.1.3 IllegalMonitorStateException	1866
6.388.1.4 IllegalMonitorStateException	1867
6.388.1.5 IllegalMonitorStateException	1867
6.388.1.6 IllegalMonitorStateException	1867
6.388.1.7 ~IllegalMonitorStateException	1867
6.388.2 Member Function Documentation	1867
6.388.2.1 clone	1867
6.389cms::IllegalStateException Class Reference	1868
6.389.1 Detailed Description	1868
6.389.2 Constructor & Destructor Documentation	1869
6.389.2.1 IllegalStateException	1869
6.389.2.2 IllegalStateException	1869
6.389.2.3 IllegalStateException	1869
6.389.2.4 IllegalStateException	1869
6.389.2.5 ~IllegalStateException	1869
6.390decaf::lang::exceptions::IllegalStateException Class Reference	1869
6.390.1 Constructor & Destructor Documentation	1870

6.390.1.1	IllegalStateException	1870
6.390.1.2	IllegalStateException	1870
6.390.1.3	IllegalStateException	1870
6.390.1.4	IllegalStateException	1870
6.390.1.5	IllegalStateException	1871
6.390.1.6	IllegalStateException	1871
6.390.1.7	~IllegalStateException	1871
6.390.2	Member Function Documentation	1871
6.390.2.1	clone	1871
6.391	decaf::lang::exceptions::IllegalThreadStateException Class Reference	1872
6.391.1	Constructor & Destructor Documentation	1872
6.391.1.1	IllegalThreadStateException	1872
6.391.1.2	IllegalThreadStateException	1872
6.391.1.3	IllegalThreadStateException	1873
6.391.1.4	IllegalThreadStateException	1873
6.391.1.5	IllegalThreadStateException	1873
6.391.1.6	IllegalThreadStateException	1873
6.391.1.7	~IllegalThreadStateException	1874
6.391.2	Member Function Documentation	1874
6.391.2.1	clone	1874
6.392	activemq::transport::inactivity::InactivityMonitor Class Reference	1874
6.392.1	Constructor & Destructor Documentation	1875
6.392.1.1	InactivityMonitor	1875
6.392.1.2	InactivityMonitor	1875
6.392.1.3	~InactivityMonitor	1875
6.392.2	Member Function Documentation	1875
6.392.2.1	close	1875
6.392.2.2	getInitialDelayTime	1876
6.392.2.3	getReadCheckTime	1876
6.392.2.4	getWriteCheckTime	1876
6.392.2.5	isKeepAliveResponseRequired	1876
6.392.2.6	onCommand	1876
6.392.2.7	oneway	1876
6.392.2.8	onException	1876
6.392.2.9	setInitialDelayTime	1877
6.392.2.10	setKeepAliveResponseRequired	1877

6.392.2.1	setReadCheckTime	1877
6.392.2.2	setWriteCheckTime	1877
6.392.3	Friends And Related Function Documentation	1877
6.392.3.1	AsyncSignalReadErrorTask	1877
6.392.3.2	AsyncWriteTask	1877
6.392.3.3	ReadChecker	1877
6.392.3.4	WriteChecker	1877
6.393	decaf::lang::exceptions::IndexOutOfBoundsException Class Reference	1877
6.393.1	Constructor & Destructor Documentation	1878
6.393.1.1	IndexOutOfBoundsException	1878
6.393.1.2	IndexOutOfBoundsException	1878
6.393.1.3	IndexOutOfBoundsException	1878
6.393.1.4	IndexOutOfBoundsException	1879
6.393.1.5	IndexOutOfBoundsException	1879
6.393.1.6	IndexOutOfBoundsException	1879
6.393.1.7	~IndexOutOfBoundsException	1879
6.393.2	Member Function Documentation	1879
6.393.2.1	clone	1879
6.394	decaf::net::Inet4Address Class Reference	1880
6.394.1	Constructor & Destructor Documentation	1881
6.394.1.1	Inet4Address	1881
6.394.1.2	Inet4Address	1881
6.394.1.3	Inet4Address	1881
6.394.1.4	~Inet4Address	1881
6.394.2	Member Function Documentation	1881
6.394.2.1	isAnyLocalAddress	1881
6.394.2.2	isLinkLocalAddress	1881
6.394.2.3	isLoopbackAddress	1882
6.394.2.4	isMCGlobal	1882
6.394.2.5	isMCLinkLocal	1882
6.394.2.6	isMCNodeLocal	1882
6.394.2.7	isMCOrgLocal	1882
6.394.2.8	isMCSiteLocal	1883
6.394.2.9	isMulticastAddress	1883
6.394.2.10	sSiteLocalAddress	1883
6.394.3	Friends And Related Function Documentation	1883

6.394.3.1 InetAddress	1883
6.395decaf::net::Inet6Address Class Reference	1883
6.395.1 Constructor & Destructor Documentation	1884
6.395.1.1 Inet6Address	1884
6.395.1.2 Inet6Address	1884
6.395.1.3 Inet6Address	1884
6.395.1.4 ~Inet6Address	1884
6.395.2 Friends And Related Function Documentation	1884
6.395.2.1 InetAddress	1884
6.396decaf::net::InetAddress Class Reference	1884
6.396.1 Detailed Description	1886
6.396.2 Constructor & Destructor Documentation	1887
6.396.2.1 InetAddress	1887
6.396.2.2 InetAddress	1887
6.396.2.3 InetAddress	1887
6.396.2.4 ~InetAddress	1887
6.396.3 Member Function Documentation	1887
6.396.3.1 bytesToInt	1887
6.396.3.2 getAddress	1887
6.396.3.3 getByAddress	1887
6.396.3.4 getByAddress	1888
6.396.3.5 getHostAddress	1888
6.396.3.6 getHostName	1888
6.396.3.7 getLocalHost	1888
6.396.3.8 isAnyLocalAddress	1889
6.396.3.9 isLinkLocalAddress	1889
6.396.3.10sLoopbackAddress	1889
6.396.3.11sMCGlobal	1889
6.396.3.12sMCLinkLocal	1889
6.396.3.13sMCNodeLocal	1890
6.396.3.14sMCOrgLocal	1890
6.396.3.15sMCSiteLocal	1890
6.396.3.16sMulticastAddress	1890
6.396.3.17sSiteLocalAddress	1890
6.396.3.18oString	1891
6.396.4 Field Documentation	1891

6.396.4.1 addressBytes	1891
6.396.4.2 ANY	1891
6.396.4.3 anyBytes	1891
6.396.4.4 hostname	1891
6.396.4.5 LOOPBACK	1891
6.396.4.6 loopbackBytes	1891
6.396.4.7 reached	1891
6.397decaf::net::InetSocketAddress Class Reference	1891
6.397.1 Constructor & Destructor Documentation	1892
6.397.1.1 InetSocketAddress	1892
6.397.1.2 ~InetSocketAddress	1892
6.398inflate_state Struct Reference	1892
6.398.1 Field Documentation	1893
6.398.1.1 back	1893
6.398.1.2 bits	1893
6.398.1.3 check	1893
6.398.1.4 codes	1893
6.398.1.5 distbits	1893
6.398.1.6 distcode	1893
6.398.1.7 dmax	1893
6.398.1.8 extra	1893
6.398.1.9 flags	1893
6.398.1.10have	1893
6.398.1.11havedict	1893
6.398.1.12head	1893
6.398.1.13hold	1893
6.398.1.14ast	1893
6.398.1.15enbits	1893
6.398.1.16encode	1893
6.398.1.17length	1893
6.398.1.18ens	1893
6.398.1.19mode	1893
6.398.1.20hcode	1893
6.398.1.21ndist	1893
6.398.1.22next	1893
6.398.1.23hlen	1893

6.398.1.24offset	1893
6.398.1.25sane	1893
6.398.1.26total	1893
6.398.1.27was	1893
6.398.1.28vbits	1893
6.398.1.29whave	1893
6.398.1.30window	1893
6.398.1.31wnext	1893
6.398.1.32work	1893
6.398.1.33wrap	1893
6.398.1.34wsiz	1893
6.399defac::util::zip::Inflater Class Reference	1894
6.399.1 Detailed Description	1895
6.399.2 Constructor & Destructor Documentation	1896
6.399.2.1 Inflater	1896
6.399.2.2 Inflater	1896
6.399.2.3 ~Inflater	1896
6.399.3 Member Function Documentation	1896
6.399.3.1 end	1896
6.399.3.2 finish	1896
6.399.3.3 finished	1896
6.399.3.4 getAdler	1896
6.399.3.5 getBytesRead	1897
6.399.3.6 getBytesWritten	1897
6.399.3.7 getRemaining	1897
6.399.3.8 inflate	1897
6.399.3.9 inflate	1898
6.399.3.10inflate	1898
6.399.3.11needsDictionary	1899
6.399.3.12needsInput	1899
6.399.3.13reset	1899
6.399.3.14setDictionary	1899
6.399.3.15setDictionary	1900
6.399.3.16setDictionary	1900
6.399.3.17setInput	1901
6.399.3.18setInput	1901

6.399.3.19	setInput	1901
6.400	decaf::util::zip::InflaterInputStream Class Reference	1902
6.400.1	Detailed Description	1904
6.400.2	Constructor & Destructor Documentation	1904
6.400.2.1	InflaterInputStream	1904
6.400.2.2	InflaterInputStream	1905
6.400.2.3	InflaterInputStream	1905
6.400.2.4	~InflaterInputStream	1906
6.400.3	Member Function Documentation	1906
6.400.3.1	available	1906
6.400.3.2	close	1906
6.400.3.3	doReadArrayBounded	1906
6.400.3.4	doReadByte	1906
6.400.3.5	fill	1907
6.400.3.6	mark	1907
6.400.3.7	markSupported	1907
6.400.3.8	reset	1907
6.400.3.9	skip	1908
6.400.4	Field Documentation	1909
6.400.4.1	atEOF	1909
6.400.4.2	buff	1909
6.400.4.3	DEFAULT_BUFFER_SIZE	1909
6.400.4.4	inflater	1909
6.400.4.5	length	1909
6.400.4.6	ownInflater	1909
6.401	decaf::io::InputStream Class Reference	1909
6.401.1	Detailed Description	1911
6.401.2	Constructor & Destructor Documentation	1911
6.401.2.1	InputStream	1911
6.401.2.2	~InputStream	1911
6.401.3	Member Function Documentation	1911
6.401.3.1	available	1911
6.401.3.2	close	1912
6.401.3.3	doReadArray	1912
6.401.3.4	doReadArrayBounded	1912
6.401.3.5	doReadByte	1912

6.401.3.6 lock	1913
6.401.3.7 mark	1913
6.401.3.8 markSupported	1913
6.401.3.9 notify	1914
6.401.3.10 notifyAll	1914
6.401.3.11 read	1914
6.401.3.12 read	1915
6.401.3.13 read	1915
6.401.3.14 reset	1916
6.401.3.15 skip	1917
6.401.3.16 toString	1917
6.401.3.17 tryLock	1917
6.401.3.18 unlock	1918
6.401.3.19 wait	1918
6.401.3.20 wait	1918
6.401.3.21 wait	1919
6.402 decaf::io::InputStreamReader Class Reference	1919
6.402.1 Detailed Description	1920
6.402.2 Constructor & Destructor Documentation	1920
6.402.2.1 InputStreamReader	1920
6.402.2.2 ~InputStreamReader	1921
6.402.3 Member Function Documentation	1921
6.402.3.1 checkClosed	1921
6.402.3.2 close	1921
6.402.3.3 doReadArrayBounded	1921
6.402.3.4 ready	1921
6.403 decaf::internal::nio::IntArrayBuffer Class Reference	1921
6.403.1 Constructor & Destructor Documentation	1925
6.403.1.1 IntArrayBuffer	1925
6.403.1.2 IntArrayBuffer	1925
6.403.1.3 IntArrayBuffer	1925
6.403.1.4 IntArrayBuffer	1926
6.403.1.5 ~IntArrayBuffer	1926
6.403.2 Member Function Documentation	1926
6.403.2.1 array	1926
6.403.2.2 arrayOffset	1927

6.403.2.3	asReadOnlyBuffer	1927
6.403.2.4	compact	1927
6.403.2.5	duplicate	1928
6.403.2.6	get	1928
6.403.2.7	get	1928
6.403.2.8	hasArray	1929
6.403.2.9	isReadOnly	1929
6.403.2.10	put	1929
6.403.2.11	put	1930
6.403.2.12	setReadOnly	1930
6.403.2.13	slice	1930
6.404	decaf::nio::IntBuffer Class Reference	1931
6.404.1	Detailed Description	1933
6.404.2	Constructor & Destructor Documentation	1933
6.404.2.1	IntBuffer	1933
6.404.2.2	~IntBuffer	1933
6.404.3	Member Function Documentation	1933
6.404.3.1	allocate	1933
6.404.3.2	array	1934
6.404.3.3	arrayOffset	1934
6.404.3.4	asReadOnlyBuffer	1934
6.404.3.5	compact	1935
6.404.3.6	compareTo	1935
6.404.3.7	duplicate	1935
6.404.3.8	equals	1936
6.404.3.9	get	1936
6.404.3.10	get	1936
6.404.3.11	get	1936
6.404.3.12	get	1937
6.404.3.13	hasArray	1937
6.404.3.14	operator<	1938
6.404.3.15	operator==	1938
6.404.3.16	put	1938
6.404.3.17	put	1938
6.404.3.18	put	1938
6.404.3.19	put	1939

6.404.3.20put	1939
6.404.3.21slice	1940
6.404.3.22toString	1940
6.404.3.23wrap	1940
6.404.3.24wrap	1941
6.405decaf::lang::Integer Class Reference	1941
6.405.1 Constructor & Destructor Documentation	1944
6.405.1.1 Integer	1944
6.405.1.2 Integer	1945
6.405.1.3 ~Integer	1945
6.405.2 Member Function Documentation	1945
6.405.2.1 bitCount	1945
6.405.2.2 byteValue	1945
6.405.2.3 compareTo	1945
6.405.2.4 compareTo	1946
6.405.2.5 decode	1946
6.405.2.6 doubleValue	1946
6.405.2.7 equals	1946
6.405.2.8 equals	1947
6.405.2.9 float Value	1947
6.405.2.10highestOneBit	1947
6.405.2.11int Value	1947
6.405.2.12long Value	1948
6.405.2.13lowestOneBit	1948
6.405.2.14numberOfLeadingZeros	1948
6.405.2.15numberOfTrailingZeros	1948
6.405.2.16operator<	1949
6.405.2.17operator<	1949
6.405.2.18operator==	1949
6.405.2.19operator==	1950
6.405.2.20parseInt	1950
6.405.2.21parseInt	1950
6.405.2.22reverse	1951
6.405.2.23reverseBytes	1951
6.405.2.24rotateLeft	1951
6.405.2.25rotateRight	1952

6.405.2.26	shortValue	1952
6.405.2.27	signum	1952
6.405.2.28	oBinaryString	1953
6.405.2.29	oHexString	1953
6.405.2.30	oOctalString	1953
6.405.2.31	toString	1954
6.405.2.32	oString	1954
6.405.2.33	oString	1954
6.405.2.34	valueOf	1955
6.405.2.35	valueOf	1955
6.405.2.36	valueOf	1955
6.405.3	Field Documentation	1956
6.405.3.1	MAX_VALUE	1956
6.405.3.2	MIN_VALUE	1956
6.405.3.3	SIZE	1956
6.406	activemq::commands::IntegerResponse Class Reference	1956
6.406.1	Constructor & Destructor Documentation	1957
6.406.1.1	IntegerResponse	1957
6.406.1.2	~IntegerResponse	1957
6.406.2	Member Function Documentation	1957
6.406.2.1	cloneDataStructure	1957
6.406.2.2	copyDataStructure	1957
6.406.2.3	equals	1957
6.406.2.4	getDataStructureType	1958
6.406.2.5	getResult	1958
6.406.2.6	setResult	1958
6.406.2.7	toString	1958
6.406.3	Field Documentation	1958
6.406.3.1	ID_INTEGERRESPONSE	1958
6.406.3.2	result	1958
6.407	activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller Class Reference	1958
6.407.1	Detailed Description	1959
6.407.2	Constructor & Destructor Documentation	1960
6.407.2.1	IntegerResponseMarshaller	1960
6.407.2.2	~IntegerResponseMarshaller	1960
6.407.3	Member Function Documentation	1960

6.407.3.1	createObject	1960
6.407.3.2	getDataStructureType	1960
6.407.3.3	looseMarshal	1960
6.407.3.4	looseUnmarshal	1961
6.407.3.5	tightMarshal1	1961
6.407.3.6	tightMarshal2	1962
6.407.3.7	tightUnmarshal	1962
6.408	activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	Class
	Reference	1962
6.408.1	Detailed Description	1963
6.408.2	Constructor & Destructor Documentation	1964
6.408.2.1	IntegerResponseMarshaller	1964
6.408.2.2	~IntegerResponseMarshaller	1964
6.408.3	Member Function Documentation	1964
6.408.3.1	createObject	1964
6.408.3.2	getDataStructureType	1964
6.408.3.3	looseMarshal	1964
6.408.3.4	looseUnmarshal	1965
6.408.3.5	tightMarshal1	1965
6.408.3.6	tightMarshal2	1966
6.408.3.7	tightUnmarshal	1966
6.409	activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	Class
	Reference	1966
6.409.1	Detailed Description	1967
6.409.2	Constructor & Destructor Documentation	1968
6.409.2.1	IntegerResponseMarshaller	1968
6.409.2.2	~IntegerResponseMarshaller	1968
6.409.3	Member Function Documentation	1968
6.409.3.1	createObject	1968
6.409.3.2	getDataStructureType	1968
6.409.3.3	looseMarshal	1968
6.409.3.4	looseUnmarshal	1969
6.409.3.5	tightMarshal1	1969
6.409.3.6	tightMarshal2	1970
6.409.3.7	tightUnmarshal	1970
6.410	activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	Class
	Reference	1970

6.410.1 Detailed Description	1971
6.410.2 Constructor & Destructor Documentation	1972
6.410.2.1 IntegerResponseMarshaller	1972
6.410.2.2 ~IntegerResponseMarshaller	1972
6.410.3 Member Function Documentation	1972
6.410.3.1 createObject	1972
6.410.3.2 getDataStructureType	1972
6.410.3.3 looseMarshal	1972
6.410.3.4 looseUnmarshal	1973
6.410.3.5 tightMarshal1	1973
6.410.3.6 tightMarshal2	1974
6.410.3.7 tightUnmarshal	1974
6.411activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller Class	
Reference	1974
6.411.1 Detailed Description	1975
6.411.2 Constructor & Destructor Documentation	1976
6.411.2.1 IntegerResponseMarshaller	1976
6.411.2.2 ~IntegerResponseMarshaller	1976
6.411.3 Member Function Documentation	1976
6.411.3.1 createObject	1976
6.411.3.2 getDataStructureType	1976
6.411.3.3 looseMarshal	1976
6.411.3.4 looseUnmarshal	1977
6.411.3.5 tightMarshal1	1977
6.411.3.6 tightMarshal2	1978
6.411.3.7 tightUnmarshal	1978
6.412activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller Class	
Reference	1978
6.412.1 Detailed Description	1979
6.412.2 Constructor & Destructor Documentation	1980
6.412.2.1 IntegerResponseMarshaller	1980
6.412.2.2 ~IntegerResponseMarshaller	1980
6.412.3 Member Function Documentation	1980
6.412.3.1 createObject	1980
6.412.3.2 getDataStructureType	1980
6.412.3.3 looseMarshal	1980
6.412.3.4 looseUnmarshal	1981

6.412.3.5 tightMarshal	1981
6.412.3.6 tightMarshal2	1982
6.412.3.7 tightUnmarshal	1982
6.413internal_state Struct Reference	1982
6.413.1 Field Documentation	1985
6.413.1.1 bi_buf	1985
6.413.1.2 bi_valid	1985
6.413.1.3 bl_count	1985
6.413.1.4 bl_desc	1985
6.413.1.5 bl_tree	1985
6.413.1.6 block_start	1985
6.413.1.7 d_buf	1985
6.413.1.8 d_desc	1985
6.413.1.9 depth	1985
6.413.1.10 dummy	1985
6.413.1.11 dyn_dtree	1985
6.413.1.12 dyn_ltree	1985
6.413.1.13 good_match	1985
6.413.1.14 gzhead	1985
6.413.1.15 gzindex	1985
6.413.1.16 hash_bits	1985
6.413.1.17 hash_mask	1985
6.413.1.18 hash_shift	1985
6.413.1.19 hash_size	1985
6.413.1.20 head	1985
6.413.1.21 heap	1985
6.413.1.22 heap_len	1985
6.413.1.23 heap_max	1985
6.413.1.24 high_water	1985
6.413.1.25 ins_h	1985
6.413.1.26 _buf	1985
6.413.1.27 _desc	1985
6.413.1.28 ast_eob_len	1985
6.413.1.29 ast_flush	1985
6.413.1.30 ast_lit	1985
6.413.1.31 level	1985

6.413.1.32	lit_bufsize	1985
6.413.1.33	lookahead	1985
6.413.1.34	match_available	1985
6.413.1.35	match_length	1985
6.413.1.36	match_start	1985
6.413.1.37	matches	1985
6.413.1.38	max_chain_length	1985
6.413.1.39	max_lazy_match	1985
6.413.1.40	method	1985
6.413.1.41	nice_match	1985
6.413.1.42	opt_len	1985
6.413.1.43	pending	1985
6.413.1.44	pending_buf	1985
6.413.1.45	pending_buf_size	1985
6.413.1.46	pending_out	1985
6.413.1.47	prev	1985
6.413.1.48	prev_length	1985
6.413.1.49	prev_match	1985
6.413.1.50	static_len	1985
6.413.1.51	status	1985
6.413.1.52	strategy	1985
6.413.1.53	trm	1985
6.413.1.54	trstart	1985
6.413.1.55	w_bits	1985
6.413.1.56	w_mask	1985
6.413.1.57	w_size	1985
6.413.1.58	window	1985
6.413.1.59	window_size	1985
6.413.1.60	wrap	1985
6.414	activemq::transport::mock::InternalCommandListener Class Reference	1986
6.414.1	Detailed Description	1986
6.414.2	Constructor & Destructor Documentation	1987
6.414.2.1	InternalCommandListener	1987
6.414.2.2	~InternalCommandListener	1987
6.414.3	Member Function Documentation	1987
6.414.3.1	onCommand	1987

6.414.3.2 run	1987
6.414.3.3 setResponseBuilder	1987
6.414.3.4 setTransport	1987
6.415decaf::lang::exceptions::InterruptedException Class Reference	1987
6.415.1 Constructor & Destructor Documentation	1988
6.415.1.1 InterruptedException	1988
6.415.1.2 InterruptedException	1988
6.415.1.3 InterruptedException	1988
6.415.1.4 InterruptedException	1988
6.415.1.5 InterruptedException	1989
6.415.1.6 InterruptedException	1989
6.415.1.7 ~InterruptedException	1989
6.415.2 Member Function Documentation	1989
6.415.2.1 clone	1989
6.416decaf::io::InterruptedException Class Reference	1990
6.416.1 Constructor & Destructor Documentation	1990
6.416.1.1 InterruptedException	1990
6.416.1.2 InterruptedException	1991
6.416.1.3 InterruptedException	1991
6.416.1.4 InterruptedException	1991
6.416.1.5 InterruptedException	1991
6.416.1.6 InterruptedException	1991
6.416.1.7 ~InterruptedException	1992
6.416.2 Member Function Documentation	1992
6.416.2.1 clone	1992
6.417cms::InvalidClientIdException Class Reference	1992
6.417.1 Detailed Description	1993
6.417.2 Constructor & Destructor Documentation	1993
6.417.2.1 InvalidClientIdException	1993
6.417.2.2 InvalidClientIdException	1993
6.417.2.3 InvalidClientIdException	1993
6.417.2.4 InvalidClientIdException	1993
6.417.2.5 ~InvalidClientIdException	1993
6.418cms::InvalidDestinationException Class Reference	1993
6.418.1 Detailed Description	1994
6.418.2 Constructor & Destructor Documentation	1994

6.418.2.1 InvalidDestinationException	1994
6.418.2.2 InvalidDestinationException	1994
6.418.2.3 InvalidDestinationException	1994
6.418.2.4 InvalidDestinationException	1994
6.418.2.5 ~InvalidDestinationException	1994
6.419decaf::security::InvalidKeyException Class Reference	1994
6.419.1 Constructor & Destructor Documentation	1995
6.419.1.1 InvalidKeyException	1995
6.419.1.2 InvalidKeyException	1995
6.419.1.3 InvalidKeyException	1995
6.419.1.4 InvalidKeyException	1995
6.419.1.5 InvalidKeyException	1996
6.419.1.6 InvalidKeyException	1996
6.419.1.7 ~InvalidKeyException	1996
6.419.2 Member Function Documentation	1996
6.419.2.1 clone	1996
6.420decaf::nio::InvalidMarkException Class Reference	1997
6.420.1 Constructor & Destructor Documentation	1997
6.420.1.1 InvalidMarkException	1997
6.420.1.2 InvalidMarkException	1998
6.420.1.3 InvalidMarkException	1998
6.420.1.4 InvalidMarkException	1998
6.420.1.5 InvalidMarkException	1998
6.420.1.6 InvalidMarkException	1998
6.420.1.7 ~InvalidMarkException	1999
6.420.2 Member Function Documentation	1999
6.420.2.1 clone	1999
6.421cms::InvalidSelectorException Class Reference	1999
6.421.1 Detailed Description	2000
6.421.2 Constructor & Destructor Documentation	2000
6.421.2.1 InvalidSelectorException	2000
6.421.2.2 InvalidSelectorException	2000
6.421.2.3 InvalidSelectorException	2000
6.421.2.4 InvalidSelectorException	2000
6.421.2.5 ~InvalidSelectorException	2000
6.422decaf::lang::exceptions::InvalidStateException Class Reference	2000

6.422.1 Constructor & Destructor Documentation	2001
6.422.1.1 InvalidStateException	2001
6.422.1.2 InvalidStateException	2001
6.422.1.3 InvalidStateException	2001
6.422.1.4 InvalidStateException	2001
6.422.1.5 InvalidStateException	2002
6.422.1.6 InvalidStateException	2002
6.422.1.7 ~InvalidStateException	2002
6.422.2 Member Function Documentation	2002
6.422.2.1 clone	2002
6.423decaf::io::IOException Class Reference	2003
6.423.1 Constructor & Destructor Documentation	2003
6.423.1.1 IOException	2003
6.423.1.2 IOException	2004
6.423.1.3 IOException	2004
6.423.1.4 IOException	2004
6.423.1.5 IOException	2004
6.423.1.6 IOException	2004
6.423.1.7 ~IOException	2005
6.423.2 Member Function Documentation	2005
6.423.2.1 clone	2005
6.424activemq::transport::IOTransport Class Reference	2005
6.424.1 Detailed Description	2007
6.424.2 Constructor & Destructor Documentation	2007
6.424.2.1 IOTransport	2007
6.424.2.2 IOTransport	2007
6.424.2.3 ~IOTransport	2007
6.424.3 Member Function Documentation	2007
6.424.3.1 close	2007
6.424.3.2 getRemoteAddress	2008
6.424.3.3 getTransportListener	2008
6.424.3.4 isClosed	2008
6.424.3.5 isConnected	2008
6.424.3.6 isFaultTolerant	2008
6.424.3.7 narrow	2009
6.424.3.8 oneway	2009

6.424.3.9 reconnect	2009
6.424.3.10 request	2009
6.424.3.11 request	2010
6.424.3.12 run	2010
6.424.3.13 setInputStream	2010
6.424.3.14 setOutputStream	2010
6.424.3.15 setTransportListener	2011
6.424.3.16 setWireFormat	2011
6.424.3.17 start	2011
6.424.3.18 stop	2011
6.425 decaf::lang::Iterable< E > Class Template Reference	2011
6.425.1 Detailed Description	2012
6.425.2 Constructor & Destructor Documentation	2012
6.425.2.1 ~Iterable	2012
6.425.3 Member Function Documentation	2012
6.425.3.1 iterator	2012
6.425.3.2 iterator	2012
6.426 decaf::util::Iterator< T > Class Template Reference	2012
6.426.1 Detailed Description	2013
6.426.2 Constructor & Destructor Documentation	2013
6.426.2.1 ~Iterator	2013
6.426.3 Member Function Documentation	2013
6.426.3.1 hasNext	2013
6.426.3.2 next	2013
6.426.3.3 remove	2014
6.427 activemq::commands::JournalQueueAck Class Reference	2014
6.427.1 Constructor & Destructor Documentation	2015
6.427.1.1 JournalQueueAck	2015
6.427.1.2 ~JournalQueueAck	2015
6.427.2 Member Function Documentation	2015
6.427.2.1 cloneDataStructure	2015
6.427.2.2 copyDataStructure	2015
6.427.2.3 equals	2016
6.427.2.4 getDataStructureType	2016
6.427.2.5 getDestination	2016
6.427.2.6 getDestination	2016

6.427.2.7	getMessageAck	2016
6.427.2.8	getMessageAck	2016
6.427.2.9	setDestination	2016
6.427.2.10	setMessageAck	2016
6.427.2.11	toString	2016
6.427.3	Field Documentation	2017
6.427.3.1	destination	2017
6.427.3.2	ID_JOURNALQUEUEACK	2017
6.427.3.3	messageAck	2017
6.428	activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller Class	
	Reference	2017
6.428.1	Detailed Description	2018
6.428.2	Constructor & Destructor Documentation	2018
6.428.2.1	JournalQueueAckMarshaller	2018
6.428.2.2	~JournalQueueAckMarshaller	2018
6.428.3	Member Function Documentation	2018
6.428.3.1	createObject	2018
6.428.3.2	getDataStructureType	2019
6.428.3.3	looseMarshal	2019
6.428.3.4	looseUnmarshal	2019
6.428.3.5	tightMarshal1	2020
6.428.3.6	tightMarshal2	2020
6.428.3.7	tightUnmarshal	2020
6.429	activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller Class	
	Reference	2021
6.429.1	Detailed Description	2022
6.429.2	Constructor & Destructor Documentation	2022
6.429.2.1	JournalQueueAckMarshaller	2022
6.429.2.2	~JournalQueueAckMarshaller	2022
6.429.3	Member Function Documentation	2022
6.429.3.1	createObject	2022
6.429.3.2	getDataStructureType	2022
6.429.3.3	looseMarshal	2023
6.429.3.4	looseUnmarshal	2023
6.429.3.5	tightMarshal1	2023
6.429.3.6	tightMarshal2	2024
6.429.3.7	tightUnmarshal	2024

6.430	activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	Class	
	Reference		2025
6.430.1	Detailed Description		2026
6.430.2	Constructor & Destructor Documentation		2026
6.430.2.1	JournalQueueAckMarshaller		2026
6.430.2.2	~JournalQueueAckMarshaller		2026
6.430.3	Member Function Documentation		2026
6.430.3.1	createObject		2026
6.430.3.2	getDataStructureType		2026
6.430.3.3	looseMarshal		2026
6.430.3.4	looseUnmarshal		2027
6.430.3.5	tightMarshal1		2027
6.430.3.6	tightMarshal2		2028
6.430.3.7	tightUnmarshal		2028
6.431	activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	Class	
	Reference		2028
6.431.1	Detailed Description		2029
6.431.2	Constructor & Destructor Documentation		2030
6.431.2.1	JournalQueueAckMarshaller		2030
6.431.2.2	~JournalQueueAckMarshaller		2030
6.431.3	Member Function Documentation		2030
6.431.3.1	createObject		2030
6.431.3.2	getDataStructureType		2030
6.431.3.3	looseMarshal		2030
6.431.3.4	looseUnmarshal		2031
6.431.3.5	tightMarshal1		2031
6.431.3.6	tightMarshal2		2031
6.431.3.7	tightUnmarshal		2032
6.432	activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	Class	
	Reference		2032
6.432.1	Detailed Description		2033
6.432.2	Constructor & Destructor Documentation		2034
6.432.2.1	JournalQueueAckMarshaller		2034
6.432.2.2	~JournalQueueAckMarshaller		2034
6.432.3	Member Function Documentation		2034
6.432.3.1	createObject		2034
6.432.3.2	getDataStructureType		2034

6.432.3.3 looseMarshal	2034
6.432.3.4 looseUnmarshal	2035
6.432.3.5 tightMarshal1	2035
6.432.3.6 tightMarshal2	2035
6.432.3.7 tightUnmarshal	2036
6.433activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller Class Reference	2036
6.433.1 Detailed Description	2037
6.433.2 Constructor & Destructor Documentation	2038
6.433.2.1 JournalQueueAckMarshaller	2038
6.433.2.2 ~JournalQueueAckMarshaller	2038
6.433.3 Member Function Documentation	2038
6.433.3.1 createObject	2038
6.433.3.2 getDataStructureType	2038
6.433.3.3 looseMarshal	2038
6.433.3.4 looseUnmarshal	2039
6.433.3.5 tightMarshal1	2039
6.433.3.6 tightMarshal2	2039
6.433.3.7 tightUnmarshal	2040
6.434activemq::commands::JournalTopicAck Class Reference	2040
6.434.1 Constructor & Destructor Documentation	2042
6.434.1.1 JournalTopicAck	2042
6.434.1.2 ~JournalTopicAck	2042
6.434.2 Member Function Documentation	2042
6.434.2.1 cloneDataStructure	2042
6.434.2.2 copyDataStructure	2042
6.434.2.3 equals	2042
6.434.2.4 getClientId	2043
6.434.2.5 getClientId	2043
6.434.2.6 getDataStructureType	2043
6.434.2.7 getDestination	2044
6.434.2.8 getDestination	2044
6.434.2.9 getMessageId	2044
6.434.2.10getMessageId	2044
6.434.2.11getMessageSequenceId	2044
6.434.2.12getSubscriptionName	2044
6.434.2.13getSubscriptionName	2044

6.434.2.14	getTransactionId	2044
6.434.2.15	getTransactionId	2044
6.434.2.16	setClientId	2044
6.434.2.17	setDestination	2044
6.434.2.18	setMessageId	2044
6.434.2.19	setMessageSequenceId	2044
6.434.2.20	setSubscriptionName	2044
6.434.2.21	setTransactionId	2044
6.434.2.22	toString	2044
6.434.3	Field Documentation	2045
6.434.3.1	clientId	2045
6.434.3.2	destination	2045
6.434.3.3	ID_ JOURNALTOPICACK	2045
6.434.3.4	messageId	2045
6.434.3.5	messageSequenceId	2045
6.434.3.6	subscriptionName	2045
6.434.3.7	transactionId	2045
6.435	activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller Class	
	Reference	2045
6.435.1	Detailed Description	2046
6.435.2	Constructor & Destructor Documentation	2047
6.435.2.1	JournalTopicAckMarshaller	2047
6.435.2.2	~JournalTopicAckMarshaller	2047
6.435.3	Member Function Documentation	2047
6.435.3.1	createObject	2047
6.435.3.2	getDataStructureType	2047
6.435.3.3	looseMarshal	2047
6.435.3.4	looseUnmarshal	2048
6.435.3.5	tightMarshal1	2048
6.435.3.6	tightMarshal2	2048
6.435.3.7	tightUnmarshal	2049
6.436	activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller Class	
	Reference	2049
6.436.1	Detailed Description	2050
6.436.2	Constructor & Destructor Documentation	2051
6.436.2.1	JournalTopicAckMarshaller	2051
6.436.2.2	~JournalTopicAckMarshaller	2051

6.436.3 Member Function Documentation	2051
6.436.3.1 createObject	2051
6.436.3.2 getDataStructureType	2051
6.436.3.3 looseMarshal	2051
6.436.3.4 looseUnmarshal	2052
6.436.3.5 tightMarshal1	2052
6.436.3.6 tightMarshal2	2052
6.436.3.7 tightUnmarshal	2053
6.437activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller Class	
Reference	2053
6.437.1 Detailed Description	2054
6.437.2 Constructor & Destructor Documentation	2055
6.437.2.1 JournalTopicAckMarshaller	2055
6.437.2.2 ~JournalTopicAckMarshaller	2055
6.437.3 Member Function Documentation	2055
6.437.3.1 createObject	2055
6.437.3.2 getDataStructureType	2055
6.437.3.3 looseMarshal	2055
6.437.3.4 looseUnmarshal	2056
6.437.3.5 tightMarshal1	2056
6.437.3.6 tightMarshal2	2056
6.437.3.7 tightUnmarshal	2057
6.438activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller Class	
Reference	2057
6.438.1 Detailed Description	2058
6.438.2 Constructor & Destructor Documentation	2059
6.438.2.1 JournalTopicAckMarshaller	2059
6.438.2.2 ~JournalTopicAckMarshaller	2059
6.438.3 Member Function Documentation	2059
6.438.3.1 createObject	2059
6.438.3.2 getDataStructureType	2059
6.438.3.3 looseMarshal	2059
6.438.3.4 looseUnmarshal	2060
6.438.3.5 tightMarshal1	2060
6.438.3.6 tightMarshal2	2060
6.438.3.7 tightUnmarshal	2061

6.439	activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	Class	
	Reference		2061
6.439.1	Detailed Description		2062
6.439.2	Constructor & Destructor Documentation		2063
	6.439.2.1 JournalTopicAckMarshaller		2063
	6.439.2.2 ~JournalTopicAckMarshaller		2063
6.439.3	Member Function Documentation		2063
	6.439.3.1 createObject		2063
	6.439.3.2 getDataStructureType		2063
	6.439.3.3 looseMarshal		2063
	6.439.3.4 looseUnmarshal		2064
	6.439.3.5 tightMarshal1		2064
	6.439.3.6 tightMarshal2		2064
	6.439.3.7 tightUnmarshal		2065
6.440	activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	Class	
	Reference		2065
6.440.1	Detailed Description		2066
6.440.2	Constructor & Destructor Documentation		2067
	6.440.2.1 JournalTopicAckMarshaller		2067
	6.440.2.2 ~JournalTopicAckMarshaller		2067
6.440.3	Member Function Documentation		2067
	6.440.3.1 createObject		2067
	6.440.3.2 getDataStructureType		2067
	6.440.3.3 looseMarshal		2067
	6.440.3.4 looseUnmarshal		2068
	6.440.3.5 tightMarshal1		2068
	6.440.3.6 tightMarshal2		2068
	6.440.3.7 tightUnmarshal		2069
6.441	activemq::commands::JournalTrace	Class Reference	2069
6.441.1	Constructor & Destructor Documentation		2070
	6.441.1.1 JournalTrace		2070
	6.441.1.2 ~JournalTrace		2070
6.441.2	Member Function Documentation		2070
	6.441.2.1 cloneDataStructure		2070
	6.441.2.2 copyDataStructure		2071
	6.441.2.3 equals		2071
	6.441.2.4 getDataStructureType		2071

6.441.2.5	getMessage	2071
6.441.2.6	getMessage	2071
6.441.2.7	setMessage	2071
6.441.2.8	toString	2071
6.441.3	Field Documentation	2072
6.441.3.1	ID_JOURNALTRACE	2072
6.441.3.2	message	2072
6.442	activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller Class Reference	2072
6.442.1	Detailed Description	2073
6.442.2	Constructor & Destructor Documentation	2073
6.442.2.1	JournalTraceMarshaller	2073
6.442.2.2	~JournalTraceMarshaller	2073
6.442.3	Member Function Documentation	2073
6.442.3.1	createObject	2073
6.442.3.2	getDataStructureType	2073
6.442.3.3	looseMarshal	2074
6.442.3.4	looseUnmarshal	2074
6.442.3.5	tightMarshal1	2074
6.442.3.6	tightMarshal2	2075
6.442.3.7	tightUnmarshal	2075
6.443	activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller Class Reference	2076
6.443.1	Detailed Description	2077
6.443.2	Constructor & Destructor Documentation	2077
6.443.2.1	JournalTraceMarshaller	2077
6.443.2.2	~JournalTraceMarshaller	2077
6.443.3	Member Function Documentation	2077
6.443.3.1	createObject	2077
6.443.3.2	getDataStructureType	2077
6.443.3.3	looseMarshal	2077
6.443.3.4	looseUnmarshal	2078
6.443.3.5	tightMarshal1	2078
6.443.3.6	tightMarshal2	2079
6.443.3.7	tightUnmarshal	2079
6.444	activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller Class Reference	2079

6.444.1 Detailed Description	2080
6.444.2 Constructor & Destructor Documentation	2081
6.444.2.1 JournalTraceMarshaller	2081
6.444.2.2 ~JournalTraceMarshaller	2081
6.444.3 Member Function Documentation	2081
6.444.3.1 createObject	2081
6.444.3.2 getDataStructureType	2081
6.444.3.3 looseMarshal	2081
6.444.3.4 looseUnmarshal	2082
6.444.3.5 tightMarshal1	2082
6.444.3.6 tightMarshal2	2082
6.444.3.7 tightUnmarshal	2083
6.445activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller Class Reference	2083
6.445.1 Detailed Description	2084
6.445.2 Constructor & Destructor Documentation	2085
6.445.2.1 JournalTraceMarshaller	2085
6.445.2.2 ~JournalTraceMarshaller	2085
6.445.3 Member Function Documentation	2085
6.445.3.1 createObject	2085
6.445.3.2 getDataStructureType	2085
6.445.3.3 looseMarshal	2085
6.445.3.4 looseUnmarshal	2086
6.445.3.5 tightMarshal1	2086
6.445.3.6 tightMarshal2	2086
6.445.3.7 tightUnmarshal	2087
6.446activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller Class Reference	2087
6.446.1 Detailed Description	2088
6.446.2 Constructor & Destructor Documentation	2089
6.446.2.1 JournalTraceMarshaller	2089
6.446.2.2 ~JournalTraceMarshaller	2089
6.446.3 Member Function Documentation	2089
6.446.3.1 createObject	2089
6.446.3.2 getDataStructureType	2089
6.446.3.3 looseMarshal	2089
6.446.3.4 looseUnmarshal	2090

6.446.3.5 tightMarshal1	2090
6.446.3.6 tightMarshal2	2090
6.446.3.7 tightUnmarshal	2091
6.447activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller Class Reference	2091
6.447.1 Detailed Description	2092
6.447.2 Constructor & Destructor Documentation	2093
6.447.2.1 JournalTraceMarshaller	2093
6.447.2.2 ~JournalTraceMarshaller	2093
6.447.3 Member Function Documentation	2093
6.447.3.1 createObject	2093
6.447.3.2 getDataStructureType	2093
6.447.3.3 looseMarshal	2093
6.447.3.4 looseUnmarshal	2094
6.447.3.5 tightMarshal1	2094
6.447.3.6 tightMarshal2	2094
6.447.3.7 tightUnmarshal	2095
6.448activemq::commands::JournalTransaction Class Reference	2095
6.448.1 Constructor & Destructor Documentation	2096
6.448.1.1 JournalTransaction	2096
6.448.1.2 ~JournalTransaction	2096
6.448.2 Member Function Documentation	2096
6.448.2.1 cloneDataStructure	2096
6.448.2.2 copyDataStructure	2097
6.448.2.3 equals	2097
6.448.2.4 getDataStructureType	2097
6.448.2.5 getTransactionId	2098
6.448.2.6 getTransactionId	2098
6.448.2.7 getType	2098
6.448.2.8 getWasPrepared	2098
6.448.2.9 setTransactionId	2098
6.448.2.10 setType	2098
6.448.2.11 setWasPrepared	2098
6.448.2.12 toString	2098
6.448.3 Field Documentation	2098
6.448.3.1 ID_JOURNALTRANSACTION	2098
6.448.3.2 transactionId	2098

6.448.3.3 type	2098
6.448.3.4 wasPrepared	2098
6.449activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller Class	
Reference	2099
6.449.1 Detailed Description	2100
6.449.2 Constructor & Destructor Documentation	2100
6.449.2.1 JournalTransactionMarshaller	2100
6.449.2.2 ~JournalTransactionMarshaller	2100
6.449.3 Member Function Documentation	2100
6.449.3.1 createObject	2100
6.449.3.2 getDataStructureType	2100
6.449.3.3 looseMarshal	2100
6.449.3.4 looseUnmarshal	2101
6.449.3.5 tightMarshal1	2101
6.449.3.6 tightMarshal2	2102
6.449.3.7 tightUnmarshal	2102
6.450activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller Class	
Reference	2102
6.450.1 Detailed Description	2103
6.450.2 Constructor & Destructor Documentation	2104
6.450.2.1 JournalTransactionMarshaller	2104
6.450.2.2 ~JournalTransactionMarshaller	2104
6.450.3 Member Function Documentation	2104
6.450.3.1 createObject	2104
6.450.3.2 getDataStructureType	2104
6.450.3.3 looseMarshal	2104
6.450.3.4 looseUnmarshal	2105
6.450.3.5 tightMarshal1	2105
6.450.3.6 tightMarshal2	2105
6.450.3.7 tightUnmarshal	2106
6.451activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller Class	
Reference	2106
6.451.1 Detailed Description	2107
6.451.2 Constructor & Destructor Documentation	2108
6.451.2.1 JournalTransactionMarshaller	2108
6.451.2.2 ~JournalTransactionMarshaller	2108
6.451.3 Member Function Documentation	2108

6.451.3.1 createObject	2108
6.451.3.2 getDataStructureType	2108
6.451.3.3 looseMarshal	2108
6.451.3.4 looseUnmarshal	2109
6.451.3.5 tightMarshal1	2109
6.451.3.6 tightMarshal2	2109
6.451.3.7 tightUnmarshal	2110
6.452activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller Class	
Reference	2110
6.452.1 Detailed Description	2111
6.452.2 Constructor & Destructor Documentation	2112
6.452.2.1 JournalTransactionMarshaller	2112
6.452.2.2 ~JournalTransactionMarshaller	2112
6.452.3 Member Function Documentation	2112
6.452.3.1 createObject	2112
6.452.3.2 getDataStructureType	2112
6.452.3.3 looseMarshal	2112
6.452.3.4 looseUnmarshal	2113
6.452.3.5 tightMarshal1	2113
6.452.3.6 tightMarshal2	2113
6.452.3.7 tightUnmarshal	2114
6.453activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller Class	
Reference	2114
6.453.1 Detailed Description	2115
6.453.2 Constructor & Destructor Documentation	2116
6.453.2.1 JournalTransactionMarshaller	2116
6.453.2.2 ~JournalTransactionMarshaller	2116
6.453.3 Member Function Documentation	2116
6.453.3.1 createObject	2116
6.453.3.2 getDataStructureType	2116
6.453.3.3 looseMarshal	2116
6.453.3.4 looseUnmarshal	2117
6.453.3.5 tightMarshal1	2117
6.453.3.6 tightMarshal2	2117
6.453.3.7 tightUnmarshal	2118
6.454activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller Class	
Reference	2118

6.454.1 Detailed Description	2119
6.454.2 Constructor & Destructor Documentation	2120
6.454.2.1 JournalTransactionMarshaller	2120
6.454.2.2 ~JournalTransactionMarshaller	2120
6.454.3 Member Function Documentation	2120
6.454.3.1 createObject	2120
6.454.3.2 getDataStructureType	2120
6.454.3.3 looseMarshal	2120
6.454.3.4 looseUnmarshal	2121
6.454.3.5 tightMarshal1	2121
6.454.3.6 tightMarshal2	2121
6.454.3.7 tightUnmarshal	2122
6.455activemq::commands::KeepAliveInfo Class Reference	2122
6.455.1 Constructor & Destructor Documentation	2123
6.455.1.1 KeepAliveInfo	2123
6.455.1.2 ~KeepAliveInfo	2123
6.455.2 Member Function Documentation	2123
6.455.2.1 cloneDataStructure	2123
6.455.2.2 copyDataStructure	2124
6.455.2.3 equals	2124
6.455.2.4 getDataStructureType	2124
6.455.2.5 isKeepAliveInfo	2124
6.455.2.6 toString	2124
6.455.2.7 visit	2125
6.455.3 Field Documentation	2125
6.455.3.1 ID_KEEPLIVEINFO	2125
6.456activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller Class Refer- ence	2125
6.456.1 Detailed Description	2126
6.456.2 Constructor & Destructor Documentation	2126
6.456.2.1 KeepAliveInfoMarshaller	2126
6.456.2.2 ~KeepAliveInfoMarshaller	2126
6.456.3 Member Function Documentation	2126
6.456.3.1 createObject	2126
6.456.3.2 getDataStructureType	2127
6.456.3.3 looseMarshal	2127
6.456.3.4 looseUnmarshal	2127

6.456.3.5 tightMarshal1	2128
6.456.3.6 tightMarshal2	2128
6.456.3.7 tightUnmarshal	2128
6.457activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller Class Reference	2129
6.457.1 Detailed Description	2130
6.457.2 Constructor & Destructor Documentation	2130
6.457.2.1 KeepAliveInfoMarshaller	2130
6.457.2.2 ~KeepAliveInfoMarshaller	2130
6.457.3 Member Function Documentation	2130
6.457.3.1 createObject	2130
6.457.3.2 getDataStructureType	2130
6.457.3.3 looseMarshal	2131
6.457.3.4 looseUnmarshal	2131
6.457.3.5 tightMarshal1	2131
6.457.3.6 tightMarshal2	2132
6.457.3.7 tightUnmarshal	2132
6.458activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller Class Reference	2133
6.458.1 Detailed Description	2134
6.458.2 Constructor & Destructor Documentation	2134
6.458.2.1 KeepAliveInfoMarshaller	2134
6.458.2.2 ~KeepAliveInfoMarshaller	2134
6.458.3 Member Function Documentation	2134
6.458.3.1 createObject	2134
6.458.3.2 getDataStructureType	2134
6.458.3.3 looseMarshal	2135
6.458.3.4 looseUnmarshal	2135
6.458.3.5 tightMarshal1	2135
6.458.3.6 tightMarshal2	2136
6.458.3.7 tightUnmarshal	2136
6.459activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller Class Reference	2137
6.459.1 Detailed Description	2138
6.459.2 Constructor & Destructor Documentation	2138
6.459.2.1 KeepAliveInfoMarshaller	2138
6.459.2.2 ~KeepAliveInfoMarshaller	2138

6.459.3 Member Function Documentation	2138
6.459.3.1 createObject	2138
6.459.3.2 getDataStructureType	2138
6.459.3.3 looseMarshal	2139
6.459.3.4 looseUnmarshal	2139
6.459.3.5 tightMarshal1	2139
6.459.3.6 tightMarshal2	2140
6.459.3.7 tightUnmarshal	2140
6.460activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller Class Reference	2141
6.460.1 Detailed Description	2142
6.460.2 Constructor & Destructor Documentation	2142
6.460.2.1 KeepAliveInfoMarshaller	2142
6.460.2.2 ~KeepAliveInfoMarshaller	2142
6.460.3 Member Function Documentation	2142
6.460.3.1 createObject	2142
6.460.3.2 getDataStructureType	2142
6.460.3.3 looseMarshal	2143
6.460.3.4 looseUnmarshal	2143
6.460.3.5 tightMarshal1	2143
6.460.3.6 tightMarshal2	2144
6.460.3.7 tightUnmarshal	2144
6.461activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller Class Reference	2145
6.461.1 Detailed Description	2146
6.461.2 Constructor & Destructor Documentation	2146
6.461.2.1 KeepAliveInfoMarshaller	2146
6.461.2.2 ~KeepAliveInfoMarshaller	2146
6.461.3 Member Function Documentation	2146
6.461.3.1 createObject	2146
6.461.3.2 getDataStructureType	2146
6.461.3.3 looseMarshal	2147
6.461.3.4 looseUnmarshal	2147
6.461.3.5 tightMarshal1	2147
6.461.3.6 tightMarshal2	2148
6.461.3.7 tightUnmarshal	2148
6.462decaf::security::Key Class Reference	2149

6.462.1 Detailed Description	2149
6.462.2 Constructor & Destructor Documentation	2150
6.462.2.1 ~Key	2150
6.462.3 Member Function Documentation	2150
6.462.3.1 getAlgorithm	2150
6.462.3.2 getEncoded	2150
6.462.3.3 getFormat	2150
6.463decaf::security::KeyException Class Reference	2151
6.463.1 Constructor & Destructor Documentation	2151
6.463.1.1 KeyException	2151
6.463.1.2 KeyException	2152
6.463.1.3 KeyException	2152
6.463.1.4 KeyException	2152
6.463.1.5 KeyException	2152
6.463.1.6 KeyException	2152
6.463.1.7 ~KeyException	2153
6.463.2 Member Function Documentation	2153
6.463.2.1 clone	2153
6.464decaf::security::KeyManagementException Class Reference	2153
6.464.1 Constructor & Destructor Documentation	2154
6.464.1.1 KeyManagementException	2154
6.464.1.2 KeyManagementException	2154
6.464.1.3 KeyManagementException	2154
6.464.1.4 KeyManagementException	2154
6.464.1.5 KeyManagementException	2155
6.464.1.6 KeyManagementException	2155
6.464.1.7 ~KeyManagementException	2155
6.464.2 Member Function Documentation	2155
6.464.2.1 clone	2155
6.465activemq::commands::LastPartialCommand Class Reference	2156
6.465.1 Constructor & Destructor Documentation	2157
6.465.1.1 LastPartialCommand	2157
6.465.1.2 ~LastPartialCommand	2157
6.465.2 Member Function Documentation	2157
6.465.2.1 cloneDataStructure	2157
6.465.2.2 copyDataStructure	2157

6.465.2.3 equals	2157
6.465.2.4 getDataStructureType	2157
6.465.2.5 toString	2158
6.465.3 Field Documentation	2158
6.465.3.1 ID_LASTPARTIALCOMMAND	2158
6.466activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller	
Class Reference	2158
6.466.1 Detailed Description	2159
6.466.2 Constructor & Destructor Documentation	2159
6.466.2.1 LastPartialCommandMarshaller	2159
6.466.2.2 ~LastPartialCommandMarshaller	2159
6.466.3 Member Function Documentation	2159
6.466.3.1 createObject	2159
6.466.3.2 getDataStructureType	2160
6.466.3.3 looseMarshal	2160
6.466.3.4 looseUnmarshal	2160
6.466.3.5 tightMarshal1	2161
6.466.3.6 tightMarshal2	2161
6.466.3.7 tightUnmarshal	2162
6.467activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	
Class Reference	2162
6.467.1 Detailed Description	2163
6.467.2 Constructor & Destructor Documentation	2163
6.467.2.1 LastPartialCommandMarshaller	2163
6.467.2.2 ~LastPartialCommandMarshaller	2163
6.467.3 Member Function Documentation	2163
6.467.3.1 createObject	2163
6.467.3.2 getDataStructureType	2164
6.467.3.3 looseMarshal	2164
6.467.3.4 looseUnmarshal	2164
6.467.3.5 tightMarshal1	2165
6.467.3.6 tightMarshal2	2165
6.467.3.7 tightUnmarshal	2165
6.468activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	
Class Reference	2166
6.468.1 Detailed Description	2167
6.468.2 Constructor & Destructor Documentation	2167

6.468.2.1 LastPartialCommandMarshaller	2167
6.468.2.2 ~LastPartialCommandMarshaller	2167
6.468.3 Member Function Documentation	2167
6.468.3.1 createObject	2167
6.468.3.2 getDataStructureType	2167
6.468.3.3 looseMarshal	2168
6.468.3.4 looseUnmarshal	2168
6.468.3.5 tightMarshal1	2169
6.468.3.6 tightMarshal2	2169
6.468.3.7 tightUnmarshal	2169
6.469activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	
Class Reference	2170
6.469.1 Detailed Description	2171
6.469.2 Constructor & Destructor Documentation	2171
6.469.2.1 LastPartialCommandMarshaller	2171
6.469.2.2 ~LastPartialCommandMarshaller	2171
6.469.3 Member Function Documentation	2171
6.469.3.1 createObject	2171
6.469.3.2 getDataStructureType	2171
6.469.3.3 looseMarshal	2172
6.469.3.4 looseUnmarshal	2172
6.469.3.5 tightMarshal1	2173
6.469.3.6 tightMarshal2	2173
6.469.3.7 tightUnmarshal	2173
6.470activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	
Class Reference	2174
6.470.1 Detailed Description	2175
6.470.2 Constructor & Destructor Documentation	2175
6.470.2.1 LastPartialCommandMarshaller	2175
6.470.2.2 ~LastPartialCommandMarshaller	2175
6.470.3 Member Function Documentation	2175
6.470.3.1 createObject	2175
6.470.3.2 getDataStructureType	2175
6.470.3.3 looseMarshal	2176
6.470.3.4 looseUnmarshal	2176
6.470.3.5 tightMarshal1	2177
6.470.3.6 tightMarshal2	2177

6.470.3.7 tightUnmarshal	2177
6.471activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	
Class Reference	2178
6.471.1 Detailed Description	2179
6.471.2 Constructor & Destructor Documentation	2179
6.471.2.1 LastPartialCommandMarshaller	2179
6.471.2.2 ~LastPartialCommandMarshaller	2179
6.471.3 Member Function Documentation	2179
6.471.3.1 createObject	2179
6.471.3.2 getDataStructureType	2179
6.471.3.3 looseMarshal	2180
6.471.3.4 looseUnmarshal	2180
6.471.3.5 tightMarshal1	2181
6.471.3.6 tightMarshal2	2181
6.471.3.7 tightUnmarshal	2181
6.472decaf::util::comparators::Less< E > Class Template Reference	2182
6.472.1 Detailed Description	2182
6.472.2 Constructor & Destructor Documentation	2183
6.472.2.1 Less	2183
6.472.2.2 ~Less	2183
6.472.3 Member Function Documentation	2183
6.472.3.1 compare	2183
6.472.3.2 operator()	2183
6.473std::less< decaf::lang::ArrayPointer< T > > Struct Template Reference	2184
6.473.1 Detailed Description	2184
6.473.2 Member Function Documentation	2184
6.473.2.1 operator()	2184
6.474std::less< decaf::lang::Pointer< T > > Struct Template Reference	2185
6.474.1 Detailed Description	2185
6.474.2 Member Function Documentation	2185
6.474.2.1 operator()	2185
6.475decaf::util::logging::Level Class Reference	2185
6.475.1 Detailed Description	2187
6.475.2 Constructor & Destructor Documentation	2187
6.475.2.1 Level	2187
6.475.2.2 ~Level	2188
6.475.3 Member Function Documentation	2188

6.475.3.1 compareTo	2188
6.475.3.2 equals	2188
6.475.3.3 getName	2188
6.475.3.4 intValue	2188
6.475.3.5 operator<	2188
6.475.3.6 operator==	2188
6.475.3.7 parse	2188
6.475.3.8 toString	2189
6.475.4 Field Documentation	2189
6.475.4.1 ALL	2189
6.475.4.2 CONFIG	2189
6.475.4.3 DEBUG	2189
6.475.4.4 FINE	2189
6.475.4.5 FINER	2189
6.475.4.6 FINEST	2190
6.475.4.7 INFO	2190
6.475.4.8 INHERIT	2190
6.475.4.9 OFF	2190
6.475.4.10 SEVERE	2190
6.475.4.11 WARNING	2190
6.476 decaf::util::List< E > Class Template Reference	2190
6.476.1 Detailed Description	2191
6.476.2 Constructor & Destructor Documentation	2192
6.476.2.1 List	2192
6.476.2.2 ~List	2192
6.476.3 Member Function Documentation	2192
6.476.3.1 add	2192
6.476.3.2 addAll	2192
6.476.3.3 get	2193
6.476.3.4 indexOf	2193
6.476.3.5 lastIndexOf	2194
6.476.3.6 listIterator	2195
6.476.3.7 listIterator	2195
6.476.3.8 listIterator	2195
6.476.3.9 listIterator	2195
6.476.3.10 remove	2196

6.476.3.1	set	2196
6.477	decaf::util::ListIterator< E > Class Template Reference	2197
6.477.1	Detailed Description	2198
6.477.2	Constructor & Destructor Documentation	2198
6.477.2.1	~ListIterator	2198
6.477.3	Member Function Documentation	2198
6.477.3.1	add	2198
6.477.3.2	hasPrevious	2199
6.477.3.3	nextIndex	2199
6.477.3.4	previous	2199
6.477.3.5	previousIndex	2199
6.477.3.6	set	2200
6.478	activemq::commands::LocalTransactionId Class Reference	2200
6.478.1	Member Typedef Documentation	2202
6.478.1.1	COMPARATOR	2202
6.478.2	Constructor & Destructor Documentation	2202
6.478.2.1	LocalTransactionId	2202
6.478.2.2	LocalTransactionId	2202
6.478.2.3	~LocalTransactionId	2202
6.478.3	Member Function Documentation	2202
6.478.3.1	cloneDataStructure	2202
6.478.3.2	compareTo	2202
6.478.3.3	copyDataStructure	2202
6.478.3.4	equals	2202
6.478.3.5	equals	2203
6.478.3.6	getConnectionId	2203
6.478.3.7	getConnectionId	2203
6.478.3.8	getDataStructureType	2203
6.478.3.9	getValue	2203
6.478.3.10	operator<	2203
6.478.3.11	operator=	2203
6.478.3.12	operator==	2203
6.478.3.13	setConnectionId	2203
6.478.3.14	setValue	2203
6.478.3.15	toString	2203
6.478.4	Field Documentation	2204

6.478.4.1	connectionId	2204
6.478.4.2	ID_LOCALTRANSACTIONID	2204
6.478.4.3	value	2204
6.479	activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller Class	
	Reference	2204
6.479.1	Detailed Description	2205
6.479.2	Constructor & Destructor Documentation	2205
6.479.2.1	LocalTransactionIdMarshaller	2205
6.479.2.2	~LocalTransactionIdMarshaller	2205
6.479.3	Member Function Documentation	2205
6.479.3.1	createObject	2205
6.479.3.2	getDataStructureType	2205
6.479.3.3	looseMarshal	2206
6.479.3.4	looseUnmarshal	2206
6.479.3.5	tightMarshal1	2206
6.479.3.6	tightMarshal2	2207
6.479.3.7	tightUnmarshal	2207
6.480	activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller Class	
	Reference	2208
6.480.1	Detailed Description	2209
6.480.2	Constructor & Destructor Documentation	2209
6.480.2.1	LocalTransactionIdMarshaller	2209
6.480.2.2	~LocalTransactionIdMarshaller	2209
6.480.3	Member Function Documentation	2209
6.480.3.1	createObject	2209
6.480.3.2	getDataStructureType	2209
6.480.3.3	looseMarshal	2210
6.480.3.4	looseUnmarshal	2210
6.480.3.5	tightMarshal1	2210
6.480.3.6	tightMarshal2	2211
6.480.3.7	tightUnmarshal	2211
6.481	activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller Class	
	Reference	2212
6.481.1	Detailed Description	2213
6.481.2	Constructor & Destructor Documentation	2213
6.481.2.1	LocalTransactionIdMarshaller	2213
6.481.2.2	~LocalTransactionIdMarshaller	2213

6.481.3 Member Function Documentation	2213
6.481.3.1 createObject	2213
6.481.3.2 getDataStructureType	2213
6.481.3.3 looseMarshal	2214
6.481.3.4 looseUnmarshal	2214
6.481.3.5 tightMarshal1	2214
6.481.3.6 tightMarshal2	2215
6.481.3.7 tightUnmarshal	2215
6.482activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller Class	
Reference	2216
6.482.1 Detailed Description	2217
6.482.2 Constructor & Destructor Documentation	2217
6.482.2.1 LocalTransactionIdMarshaller	2217
6.482.2.2 ~LocalTransactionIdMarshaller	2217
6.482.3 Member Function Documentation	2217
6.482.3.1 createObject	2217
6.482.3.2 getDataStructureType	2217
6.482.3.3 looseMarshal	2218
6.482.3.4 looseUnmarshal	2218
6.482.3.5 tightMarshal1	2218
6.482.3.6 tightMarshal2	2219
6.482.3.7 tightUnmarshal	2219
6.483activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller Class	
Reference	2220
6.483.1 Detailed Description	2221
6.483.2 Constructor & Destructor Documentation	2221
6.483.2.1 LocalTransactionIdMarshaller	2221
6.483.2.2 ~LocalTransactionIdMarshaller	2221
6.483.3 Member Function Documentation	2221
6.483.3.1 createObject	2221
6.483.3.2 getDataStructureType	2221
6.483.3.3 looseMarshal	2222
6.483.3.4 looseUnmarshal	2222
6.483.3.5 tightMarshal1	2222
6.483.3.6 tightMarshal2	2223
6.483.3.7 tightUnmarshal	2223

6.484activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	Class	
Reference		2224
6.484.1 Detailed Description		2225
6.484.2 Constructor & Destructor Documentation		2225
6.484.2.1 LocalTransactionIdMarshaller		2225
6.484.2.2 ~LocalTransactionIdMarshaller		2225
6.484.3 Member Function Documentation		2225
6.484.3.1 createObject		2225
6.484.3.2 getDataStructureType		2225
6.484.3.3 looseMarshal		2226
6.484.3.4 looseUnmarshal		2226
6.484.3.5 tightMarshal1		2226
6.484.3.6 tightMarshal2		2227
6.484.3.7 tightUnmarshal		2227
6.485decaf::util::concurrent::Lock	Class Reference	2228
6.485.1 Detailed Description		2228
6.485.2 Constructor & Destructor Documentation		2228
6.485.2.1 Lock		2228
6.485.2.2 ~Lock		2229
6.485.3 Member Function Documentation		2229
6.485.3.1 isLocked		2229
6.485.3.2 lock		2229
6.485.3.3 unlock		2229
6.486decaf::util::concurrent::locks::Lock	Class Reference	2229
6.486.1 Detailed Description		2230
6.486.2 Constructor & Destructor Documentation		2231
6.486.2.1 ~Lock		2231
6.486.3 Member Function Documentation		2231
6.486.3.1 lock		2231
6.486.3.2 lockInterruptibly		2231
6.486.3.3 newCondition		2232
6.486.3.4 tryLock		2233
6.486.3.5 tryLock		2234
6.486.3.6 unlock		2234
6.487decaf::util::concurrent::locks::LockSupport	Class Reference	2234
6.487.1 Detailed Description		2235
6.487.2 Constructor & Destructor Documentation		2236

6.487.2.1 ~LockSupport	2236
6.487.3 Member Function Documentation	2236
6.487.3.1 park	2236
6.487.3.2 parkNanos	2236
6.487.3.3 parkUntil	2237
6.487.3.4 unpark	2237
6.488decaf::util::logging::Logger Class Reference	2237
6.488.1 Detailed Description	2240
6.488.2 Constructor & Destructor Documentation	2241
6.488.2.1 Logger	2241
6.488.2.2 ~Logger	2241
6.488.3 Member Function Documentation	2241
6.488.3.1 addHandler	2241
6.488.3.2 config	2241
6.488.3.3 debug	2242
6.488.3.4 entering	2242
6.488.3.5 exiting	2242
6.488.3.6 fine	2242
6.488.3.7 finer	2243
6.488.3.8 finest	2243
6.488.3.9 getAnonymousLogger	2243
6.488.3.10getFilter	2244
6.488.3.11getHandlers	2244
6.488.3.12getLevel	2244
6.488.3.13getLogger	2244
6.488.3.14getName	2245
6.488.3.15getParent	2245
6.488.3.16getUseParentHandlers	2245
6.488.3.17info	2245
6.488.3.18sLoggable	2245
6.488.3.19log	2246
6.488.3.20log	2246
6.488.3.21log	2246
6.488.3.22log	2247
6.488.3.23removeHandler	2247
6.488.3.24setFilter	2247

6.488.3.25	setLevel	2247
6.488.3.26	setParent	2248
6.488.3.27	setUseParentHandlers	2248
6.488.3.28	severe	2248
6.488.3.29	throwing	2248
6.488.3.30	warning	2249
6.489	decaf::util::logging::LoggerHierarchy Class Reference	2249
6.489.1	Constructor & Destructor Documentation	2249
6.489.1.1	LoggerHierarchy	2249
6.489.1.2	~LoggerHierarchy	2249
6.490	activemq::io::LoggingInputStream Class Reference	2250
6.490.1	Constructor & Destructor Documentation	2250
6.490.1.1	LoggingInputStream	2250
6.490.1.2	~LoggingInputStream	2250
6.490.2	Member Function Documentation	2250
6.490.2.1	doReadArrayBounded	2250
6.490.2.2	doReadByte	2251
6.491	activemq::io::LoggingOutputStream Class Reference	2251
6.491.1	Detailed Description	2251
6.491.2	Constructor & Destructor Documentation	2251
6.491.2.1	LoggingOutputStream	2251
6.491.2.2	~LoggingOutputStream	2252
6.491.3	Member Function Documentation	2252
6.491.3.1	doWriteArrayBounded	2252
6.491.3.2	doWriteByte	2252
6.492	activemq::transport::logging::LoggingTransport Class Reference	2252
6.492.1	Detailed Description	2253
6.492.2	Constructor & Destructor Documentation	2253
6.492.2.1	LoggingTransport	2253
6.492.2.2	~LoggingTransport	2253
6.492.3	Member Function Documentation	2253
6.492.3.1	onCommand	2253
6.492.3.2	oneway	2253
6.492.3.3	request	2254
6.492.3.4	request	2254
6.493	decaf::util::logging::LogManager Class Reference	2254

6.493.1 Detailed Description	2256
6.493.2 Constructor & Destructor Documentation	2257
6.493.2.1 ~LogManager	2257
6.493.2.2 LogManager	2257
6.493.2.3 LogManager	2257
6.493.3 Member Function Documentation	2258
6.493.3.1 addLogger	2258
6.493.3.2 addPropertyChangeListener	2258
6.493.3.3 getLogger	2258
6.493.3.4 getLoggerNames	2258
6.493.3.5 getLoggerManager	2259
6.493.3.6 getProperties	2259
6.493.3.7 getProperty	2259
6.493.3.8 operator=	2259
6.493.3.9 readConfiguration	2259
6.493.3.10 readConfiguration	2260
6.493.3.11 removePropertyChangeListener	2260
6.493.3.12 reset	2260
6.493.3.13 setProperties	2260
6.493.4 Friends And Related Function Documentation	2261
6.493.4.1 decaf::lang::Runtime	2261
6.494 decaf::util::logging::LogRecord Class Reference	2261
6.494.1 Detailed Description	2262
6.494.2 Constructor & Destructor Documentation	2262
6.494.2.1 LogRecord	2262
6.494.2.2 ~LogRecord	2262
6.494.3 Member Function Documentation	2262
6.494.3.1 getLevel	2262
6.494.3.2 getLoggerName	2263
6.494.3.3 getMessage	2263
6.494.3.4 getSourceFile	2263
6.494.3.5 getSourceFunction	2263
6.494.3.6 getSourceLine	2263
6.494.3.7 getThrown	2263
6.494.3.8 getTimestamp	2264
6.494.3.9 getTreadId	2264

6.494.3.10	setLevel	2264
6.494.3.11	setLoggerName	2264
6.494.3.12	setMessage	2264
6.494.3.13	setSourceFile	2264
6.494.3.14	setSourceFunction	2265
6.494.3.15	setSourceLine	2265
6.494.3.16	setThrown	2265
6.494.3.17	setTimestamp	2265
6.494.3.18	setTreadId	2265
6.495	decaf::util::logging::LogWriter Class Reference	2266
6.495.1	Constructor & Destructor Documentation	2266
6.495.1.1	LogWriter	2266
6.495.1.2	~LogWriter	2266
6.495.2	Member Function Documentation	2266
6.495.2.1	destroy	2266
6.495.2.2	getInstance	2266
6.495.2.3	log	2267
6.495.2.4	log	2267
6.495.2.5	returnInstance	2267
6.496	decaf::lang::Long Class Reference	2267
6.496.1	Constructor & Destructor Documentation	2270
6.496.1.1	Long	2270
6.496.1.2	Long	2270
6.496.1.3	~Long	2271
6.496.2	Member Function Documentation	2271
6.496.2.1	bitCount	2271
6.496.2.2	byteValue	2271
6.496.2.3	compareTo	2271
6.496.2.4	compareTo	2271
6.496.2.5	decode	2272
6.496.2.6	doubleValue	2272
6.496.2.7	equals	2272
6.496.2.8	equals	2273
6.496.2.9	float Value	2273
6.496.2.10	highestOneBit	2273
6.496.2.11	int Value	2273

6.496.2.12	longValue	2273
6.496.2.13	lowestOneBit	2274
6.496.2.14	numberOfLeadingZeros	2274
6.496.2.15	numberOfTrailingZeros	2274
6.496.2.16	operator<	2275
6.496.2.17	operator<	2275
6.496.2.18	operator==	2275
6.496.2.19	operator==	2276
6.496.2.20	parseLong	2276
6.496.2.21	parseLong	2276
6.496.2.22	reverse	2277
6.496.2.23	reverseBytes	2277
6.496.2.24	rotateLeft	2277
6.496.2.25	rotateRight	2278
6.496.2.26	shortValue	2278
6.496.2.27	signum	2278
6.496.2.28	toBinaryString	2278
6.496.2.29	toHexString	2279
6.496.2.30	toOctalString	2279
6.496.2.31	toString	2280
6.496.2.32	toString	2280
6.496.2.33	toString	2280
6.496.2.34	valueOf	2280
6.496.2.35	valueOf	2280
6.496.2.36	valueOf	2281
6.496.3	Field Documentation	2281
6.496.3.1	MAX_VALUE	2281
6.496.3.2	MIN_VALUE	2281
6.496.3.3	SIZE	2281
6.497	decaf::internal::nio::LongArrayBuffer Class Reference	2281
6.497.1	Constructor & Destructor Documentation	2285
6.497.1.1	LongArrayBuffer	2285
6.497.1.2	LongArrayBuffer	2285
6.497.1.3	LongArrayBuffer	2286
6.497.1.4	LongArrayBuffer	2286
6.497.1.5	~LongArrayBuffer	2286

6.497.2 Member Function Documentation	2286
6.497.2.1 array	2286
6.497.2.2 arrayOffset	2287
6.497.2.3 asReadOnlyBuffer	2287
6.497.2.4 compact	2287
6.497.2.5 duplicate	2288
6.497.2.6 get	2288
6.497.2.7 get	2289
6.497.2.8 hasArray	2289
6.497.2.9 isReadOnly	2289
6.497.2.10put	2289
6.497.2.11put	2290
6.497.2.12setReadOnly	2290
6.497.2.13lice	2290
6.498decaf::nio::LongBuffer Class Reference	2291
6.498.1 Detailed Description	2293
6.498.2 Constructor & Destructor Documentation	2293
6.498.2.1 LongBuffer	2293
6.498.2.2 ~LongBuffer	2294
6.498.3 Member Function Documentation	2294
6.498.3.1 allocate	2294
6.498.3.2 array	2294
6.498.3.3 arrayOffset	2294
6.498.3.4 asReadOnlyBuffer	2295
6.498.3.5 compact	2295
6.498.3.6 compareTo	2296
6.498.3.7 duplicate	2296
6.498.3.8 equals	2296
6.498.3.9 get	2296
6.498.3.10get	2296
6.498.3.11get	2297
6.498.3.12get	2297
6.498.3.13hasArray	2298
6.498.3.14operator<	2298
6.498.3.15operator==	2298
6.498.3.16put	2298

6.498.3.17put	2298
6.498.3.18put	2299
6.498.3.19put	2299
6.498.3.20put	2300
6.498.3.21slice	2300
6.498.3.22toString	2301
6.498.3.23wrap	2301
6.498.3.24wrap	2301
6.499activemq::util::LongSequenceGenerator Class Reference	2302
6.499.1 Detailed Description	2302
6.499.2 Constructor & Destructor Documentation	2302
6.499.2.1 LongSequenceGenerator	2302
6.499.2.2 ~LongSequenceGenerator	2302
6.499.3 Member Function Documentation	2302
6.499.3.1 getLastSequenceId	2302
6.499.3.2 getNextSequenceId	2302
6.500decaf::net::MalformedURLException Class Reference	2303
6.500.1 Constructor & Destructor Documentation	2303
6.500.1.1 MalformedURLException	2303
6.500.1.2 MalformedURLException	2304
6.500.1.3 MalformedURLException	2304
6.500.1.4 MalformedURLException	2304
6.500.1.5 MalformedURLException	2304
6.500.1.6 MalformedURLException	2305
6.500.1.7 ~MalformedURLException	2305
6.500.2 Member Function Documentation	2305
6.500.2.1 clone	2305
6.501decaf::util::Map< K, V, COMPARATOR > Class Template Reference	2305
6.501.1 Detailed Description	2307
6.501.2 Constructor & Destructor Documentation	2307
6.501.2.1 Map	2307
6.501.2.2 ~Map	2307
6.501.3 Member Function Documentation	2307
6.501.3.1 clear	2307
6.501.3.2 containsKey	2308
6.501.3.3 containsValue	2309

6.501.3.4 copy	2309
6.501.3.5 equals	2310
6.501.3.6 get	2310
6.501.3.7 get	2311
6.501.3.8 isEmpty	2312
6.501.3.9 keySet	2313
6.501.3.10put	2314
6.501.3.11putAll	2314
6.501.3.12remove	2315
6.501.3.13size	2316
6.501.3.14values	2317
6.502cms::MapMessage Class Reference	2318
6.502.1 Detailed Description	2320
6.502.2 Constructor & Destructor Documentation	2320
6.502.2.1 ~MapMessage	2320
6.502.3 Member Function Documentation	2320
6.502.3.1 getBoolean	2320
6.502.3.2 getByte	2321
6.502.3.3 getBytes	2321
6.502.3.4 getChar	2321
6.502.3.5 getDouble	2322
6.502.3.6 getFloat	2322
6.502.3.7 getInt	2322
6.502.3.8 getLong	2322
6.502.3.9 getMapNames	2323
6.502.3.10getShort	2323
6.502.3.11getString	2323
6.502.3.12itemExists	2324
6.502.3.13setBoolean	2324
6.502.3.14setByte	2324
6.502.3.15setBytes	2325
6.502.3.16setChar	2325
6.502.3.17setDouble	2325
6.502.3.18setFloat	2326
6.502.3.19setInt	2326
6.502.3.20setLong	2326

6.502.3.2	setShort	2327
6.502.3.2	setString	2327
6.503	decaf::util::logging::MarkBlockLogger Class Reference	2327
6.503.1	Detailed Description	2328
6.503.2	Constructor & Destructor Documentation	2328
6.503.2.1	MarkBlockLogger	2328
6.503.2.2	~MarkBlockLogger	2328
6.504	activemq::wireformat::MarshalAware Class Reference	2328
6.504.1	Constructor & Destructor Documentation	2329
6.504.1.1	~MarshalAware	2329
6.504.2	Member Function Documentation	2329
6.504.2.1	afterMarshal	2329
6.504.2.2	afterUnmarshal	2329
6.504.2.3	beforeMarshal	2329
6.504.2.4	beforeUnmarshal	2330
6.504.2.5	getMarshaledForm	2330
6.504.2.6	isMarshalAware	2330
6.504.2.7	setMarshaledForm	2330
6.505	activemq::wireformat::openwire::marshal::v6::MarshallerFactory Class Reference	2331
6.505.1	Detailed Description	2331
6.505.2	Constructor & Destructor Documentation	2331
6.505.2.1	~MarshallerFactory	2331
6.505.3	Member Function Documentation	2331
6.505.3.1	configure	2331
6.506	activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference	2331
6.506.1	Detailed Description	2332
6.506.2	Constructor & Destructor Documentation	2332
6.506.2.1	~MarshallerFactory	2332
6.506.3	Member Function Documentation	2332
6.506.3.1	configure	2332
6.507	activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class Reference	2332
6.507.1	Detailed Description	2332
6.507.2	Constructor & Destructor Documentation	2333
6.507.2.1	~MarshallerFactory	2333
6.507.3	Member Function Documentation	2333
6.507.3.1	configure	2333

6.508	activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class Reference . .	2333
6.508.1	Detailed Description	2333
6.508.2	Constructor & Destructor Documentation	2333
6.508.2.1	~MarshallerFactory	2333
6.508.3	Member Function Documentation	2333
6.508.3.1	configure	2333
6.509	activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference . .	2334
6.509.1	Detailed Description	2334
6.509.2	Constructor & Destructor Documentation	2334
6.509.2.1	~MarshallerFactory	2334
6.509.3	Member Function Documentation	2334
6.509.3.1	configure	2334
6.510	activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class Reference . .	2334
6.510.1	Detailed Description	2335
6.510.2	Constructor & Destructor Documentation	2335
6.510.2.1	~MarshallerFactory	2335
6.510.3	Member Function Documentation	2335
6.510.3.1	configure	2335
6.511	activemq::util::MarshallingSupport Class Reference	2335
6.511.1	Constructor & Destructor Documentation	2336
6.511.1.1	MarshallingSupport	2336
6.511.1.2	~MarshallingSupport	2336
6.511.2	Member Function Documentation	2336
6.511.2.1	asciiToModifiedUtf8	2336
6.511.2.2	modifiedUtf8ToAscii	2337
6.511.2.3	readString16	2337
6.511.2.4	readString32	2337
6.511.2.5	writeString	2338
6.511.2.6	writeString16	2338
6.511.2.7	writeString32	2339
6.512	decaf::lang::Math Class Reference	2339
6.512.1	Detailed Description	2341
6.512.2	Constructor & Destructor Documentation	2341
6.512.2.1	Math	2341
6.512.2.2	~Math	2341
6.512.3	Member Function Documentation	2341

6.512.3.1 abs	2341
6.512.3.2 abs	2342
6.512.3.3 abs	2342
6.512.3.4 abs	2342
6.512.3.5 ceil	2343
6.512.3.6 floor	2344
6.512.3.7 max	2344
6.512.3.8 max	2344
6.512.3.9 max	2345
6.512.3.10max	2345
6.512.3.11max	2345
6.512.3.12min	2346
6.512.3.13min	2346
6.512.3.14min	2346
6.512.3.15min	2347
6.512.3.16min	2347
6.512.3.17min	2347
6.512.3.18pow	2348
6.512.3.19random	2348
6.512.3.20round	2349
6.512.3.21round	2349
6.512.3.22signum	2350
6.512.3.23signum	2350
6.512.3.24qrt	2351
6.512.3.25toDegrees	2354
6.512.3.26toRadians	2354
6.512.4 Field Documentation	2354
6.512.4.1 E	2354
6.512.4.2 PI	2354
6.513activemq::util::MemoryUsage Class Reference	2354
6.513.1 Constructor & Destructor Documentation	2355
6.513.1.1 MemoryUsage	2355
6.513.1.2 MemoryUsage	2356
6.513.1.3 ~MemoryUsage	2356
6.513.2 Member Function Documentation	2356
6.513.2.1 decreaseUsage	2356

6.513.2.2 enqueueUsage	2356
6.513.2.3 getLimit	2356
6.513.2.4 getUsage	2356
6.513.2.5 increaseUsage	2357
6.513.2.6 isFull	2357
6.513.2.7 setLimit	2357
6.513.2.8 setUsage	2357
6.513.2.9 waitForSpace	2357
6.513.2.10waitForSpace	2357
6.514activemq::commands::Message Class Reference	2358
6.514.1 Constructor & Destructor Documentation	2362
6.514.1.1 Message	2362
6.514.1.2 ~Message	2362
6.514.2 Member Function Documentation	2362
6.514.2.1 afterUnmarshal	2362
6.514.2.2 beforeMarshal	2362
6.514.2.3 cloneDataStructure	2363
6.514.2.4 copyDataStructure	2363
6.514.2.5 equals	2363
6.514.2.6 getAckHandler	2364
6.514.2.7 getArrival	2364
6.514.2.8 getBrokerInTime	2364
6.514.2.9 getBrokerOutTime	2364
6.514.2.10getBrokerPath	2364
6.514.2.11getBrokerPath	2364
6.514.2.12getCluster	2364
6.514.2.13getCluster	2364
6.514.2.14getConnection	2364
6.514.2.15getContent	2365
6.514.2.16getContent	2365
6.514.2.17getCorrelationId	2365
6.514.2.18getCorrelationId	2365
6.514.2.19getDataStructure	2365
6.514.2.20getDataStructure	2365
6.514.2.21getDataStructureType	2365
6.514.2.22getDestination	2366

6.514.2.23	getDestination	2366
6.514.2.24	getExpiration	2366
6.514.2.25	getGroupID	2366
6.514.2.26	getGroupID	2366
6.514.2.27	getGroupSequence	2366
6.514.2.28	getMarshaledProperties	2366
6.514.2.29	getMarshaledProperties	2366
6.514.2.30	getMessageId	2366
6.514.2.31	getMessageId	2366
6.514.2.32	getMessageProperties	2366
6.514.2.33	getMessageProperties	2367
6.514.2.34	getOriginalDestination	2367
6.514.2.35	getOriginalDestination	2367
6.514.2.36	getOriginalTransactionId	2367
6.514.2.37	getOriginalTransactionId	2367
6.514.2.38	getPriority	2367
6.514.2.39	getProducerId	2367
6.514.2.40	getProducerId	2367
6.514.2.41	getRedeliveryCounter	2367
6.514.2.42	getReplyTo	2367
6.514.2.43	getReplyTo	2367
6.514.2.44	getSize	2367
6.514.2.45	getTargetConsumerId	2368
6.514.2.46	getTargetConsumerId	2368
6.514.2.47	getTimestamp	2368
6.514.2.48	getTransactionId	2368
6.514.2.49	getTransactionId	2368
6.514.2.50	getType	2368
6.514.2.51	getType	2368
6.514.2.52	getUserID	2368
6.514.2.53	getUserID	2368
6.514.2.54	isCompressed	2368
6.514.2.55	isDroppable	2368
6.514.2.56	isExpired	2368
6.514.2.57	isMarshalAware	2368
6.514.2.58	isMessage	2369

6.514.2.59sPersistent	2369
6.514.2.60sReadOnlyBody	2369
6.514.2.61sReadOnlyProperties	2369
6.514.2.62sRecievedByDFBridge	2369
6.514.2.63nSend	2369
6.514.2.64setAckHandler	2370
6.514.2.65setArrival	2370
6.514.2.66setBrokerInTime	2370
6.514.2.67setBrokerOutTime	2370
6.514.2.68setBrokerPath	2370
6.514.2.69setCluster	2370
6.514.2.70setCompressed	2370
6.514.2.71setConnection	2370
6.514.2.72setContent	2371
6.514.2.73setCorrelationId	2371
6.514.2.74setDataStructure	2371
6.514.2.75setDestination	2371
6.514.2.76setDroppable	2371
6.514.2.77setExpiration	2371
6.514.2.78setGroupID	2371
6.514.2.79setGroupSequence	2371
6.514.2.80setMarshaledProperties	2371
6.514.2.81setMessageId	2371
6.514.2.82setOriginalDestination	2371
6.514.2.83setOriginalTransactionId	2371
6.514.2.84setPersistent	2371
6.514.2.85setPriority	2371
6.514.2.86setProducerId	2371
6.514.2.87setReadOnlyBody	2371
6.514.2.88setReadOnlyProperties	2372
6.514.2.89setRecievedByDFBridge	2372
6.514.2.90setRedeliveryCounter	2372
6.514.2.91setReplyTo	2372
6.514.2.92setTargetConsumerId	2372
6.514.2.93setTimestamp	2372
6.514.2.94setTransactionId	2372

6.514.2.95	set Type	2372
6.514.2.96	set UserID	2372
6.514.2.97	toString	2372
6.514.2.98	visit	2373
6.514.3	Field Documentation	2374
6.514.3.1	arrival	2374
6.514.3.2	brokerInTime	2374
6.514.3.3	brokerOutTime	2374
6.514.3.4	brokerPath	2374
6.514.3.5	cluster	2374
6.514.3.6	compressed	2374
6.514.3.7	connection	2374
6.514.3.8	content	2374
6.514.3.9	correlationId	2374
6.514.3.10	dataStructure	2374
6.514.3.11	DEFAULT_MESSAGE_SIZE	2374
6.514.3.12	destination	2374
6.514.3.13	droppable	2374
6.514.3.14	expiration	2374
6.514.3.15	groupId	2374
6.514.3.16	groupSequence	2374
6.514.3.17	ID_MESSAGE	2374
6.514.3.18	marshalledProperties	2374
6.514.3.19	messageId	2374
6.514.3.20	originalDestination	2374
6.514.3.21	originalTransactionId	2374
6.514.3.22	persistent	2374
6.514.3.23	priority	2374
6.514.3.24	producerId	2374
6.514.3.25	recievedByDFBridge	2374
6.514.3.26	redeliveryCounter	2374
6.514.3.27	replyTo	2374
6.514.3.28	targetConsumerId	2374
6.514.3.29	timestamp	2374
6.514.3.30	transactionId	2374
6.514.3.31	type	2374

6.514.3.32	userID	2374
6.515	cms::Message Class Reference	2375
6.515.1	Detailed Description	2378
6.515.2	Constructor & Destructor Documentation	2379
6.515.2.1	~Message	2379
6.515.3	Member Function Documentation	2379
6.515.3.1	acknowledge	2379
6.515.3.2	clearBody	2379
6.515.3.3	clearProperties	2380
6.515.3.4	clone	2380
6.515.3.5	getBooleanProperty	2380
6.515.3.6	getByteProperty	2381
6.515.3.7	getCMSCorrelationID	2381
6.515.3.8	getCMSDeliveryMode	2381
6.515.3.9	getCMSDestination	2382
6.515.3.10	getCMSExpiration	2382
6.515.3.11	getCMSMessageID	2382
6.515.3.12	getCMSPriority	2383
6.515.3.13	getCMSRedelivered	2383
6.515.3.14	getCMSReplyTo	2384
6.515.3.15	getCMSTimestamp	2384
6.515.3.16	getCMSType	2384
6.515.3.17	getDoubleProperty	2385
6.515.3.18	getFloatProperty	2385
6.515.3.19	getIntProperty	2385
6.515.3.20	getLongProperty	2386
6.515.3.21	getPropertyNames	2386
6.515.3.22	getShortProperty	2386
6.515.3.23	getStringProperty	2387
6.515.3.24	propertyExists	2387
6.515.3.25	setBooleanProperty	2387
6.515.3.26	setByteProperty	2388
6.515.3.27	setCMSCorrelationID	2388
6.515.3.28	setCMSDeliveryMode	2389
6.515.3.29	setCMSDestination	2389
6.515.3.30	setCMSExpiration	2389

6.515.3.31	setCMSMessageID	2389
6.515.3.32	setCMSPriority	2390
6.515.3.33	setCMSRedelivered	2390
6.515.3.34	setCMSReplyTo	2390
6.515.3.35	setCMSTimestamp	2391
6.515.3.36	setCMSType	2391
6.515.3.37	setDoubleProperty	2392
6.515.3.38	setFloatProperty	2392
6.515.3.39	setIntProperty	2392
6.515.3.40	setLongProperty	2393
6.515.3.41	setShortProperty	2393
6.515.3.42	setStringProperty	2393
6.516	activemq::commands::MessageAck Class Reference	2394
6.516.1	Constructor & Destructor Documentation	2395
6.516.1.1	MessageAck	2395
6.516.1.2	~MessageAck	2395
6.516.2	Member Function Documentation	2395
6.516.2.1	cloneDataStructure	2395
6.516.2.2	copyDataStructure	2395
6.516.2.3	equals	2396
6.516.2.4	getAckType	2396
6.516.2.5	getConsumerId	2396
6.516.2.6	getConsumerId	2396
6.516.2.7	getDataStructureType	2396
6.516.2.8	getDestination	2397
6.516.2.9	getDestination	2397
6.516.2.10	getFirstMessageId	2397
6.516.2.11	getFirstMessageId	2397
6.516.2.12	getLastMessageId	2397
6.516.2.13	getLastMessageId	2397
6.516.2.14	getMessageCount	2397
6.516.2.15	getTransactionId	2397
6.516.2.16	getTransactionId	2397
6.516.2.17	isMessageAck	2397
6.516.2.18	setAckType	2398
6.516.2.19	setConsumerId	2398

6.516.2.20	setDestination	2398
6.516.2.21	setFirstMessageId	2398
6.516.2.22	setLastMessageId	2398
6.516.2.23	setMessageCount	2398
6.516.2.24	setTransactionId	2398
6.516.2.25	toString	2398
6.516.2.26	visit	2398
6.516.3	Field Documentation	2399
6.516.3.1	ackType	2399
6.516.3.2	consumerId	2399
6.516.3.3	destination	2399
6.516.3.4	firstMessageId	2399
6.516.3.5	ID_MESSAGEACK	2399
6.516.3.6	lastMessageId	2399
6.516.3.7	messageCount	2399
6.516.3.8	transactionId	2399
6.517	activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller Class Reference	2399
6.517.1	Detailed Description	2400
6.517.2	Constructor & Destructor Documentation	2400
6.517.2.1	MessageAckMarshaller	2400
6.517.2.2	~MessageAckMarshaller	2400
6.517.3	Member Function Documentation	2400
6.517.3.1	createObject	2400
6.517.3.2	getDataStructureType	2401
6.517.3.3	looseMarshal	2401
6.517.3.4	looseUnmarshal	2401
6.517.3.5	tightMarshal1	2402
6.517.3.6	tightMarshal2	2402
6.517.3.7	tightUnmarshal	2402
6.518	activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller Class Reference	2403
6.518.1	Detailed Description	2404
6.518.2	Constructor & Destructor Documentation	2404
6.518.2.1	MessageAckMarshaller	2404
6.518.2.2	~MessageAckMarshaller	2404
6.518.3	Member Function Documentation	2404
6.518.3.1	createObject	2404

6.518.3.2	getDataStructureType	2404
6.518.3.3	looseMarshal	2405
6.518.3.4	looseUnmarshal	2405
6.518.3.5	tightMarshal1	2405
6.518.3.6	tightMarshal2	2406
6.518.3.7	tightUnmarshal	2406
6.519	activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller Class Reference	2407
6.519.1	Detailed Description	2408
6.519.2	Constructor & Destructor Documentation	2408
6.519.2.1	MessageAckMarshaller	2408
6.519.2.2	~MessageAckMarshaller	2408
6.519.3	Member Function Documentation	2408
6.519.3.1	createObject	2408
6.519.3.2	getDataStructureType	2408
6.519.3.3	looseMarshal	2409
6.519.3.4	looseUnmarshal	2409
6.519.3.5	tightMarshal1	2409
6.519.3.6	tightMarshal2	2410
6.519.3.7	tightUnmarshal	2410
6.520	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller Class Reference	2411
6.520.1	Detailed Description	2412
6.520.2	Constructor & Destructor Documentation	2412
6.520.2.1	MessageAckMarshaller	2412
6.520.2.2	~MessageAckMarshaller	2412
6.520.3	Member Function Documentation	2412
6.520.3.1	createObject	2412
6.520.3.2	getDataStructureType	2412
6.520.3.3	looseMarshal	2413
6.520.3.4	looseUnmarshal	2413
6.520.3.5	tightMarshal1	2413
6.520.3.6	tightMarshal2	2414
6.520.3.7	tightUnmarshal	2414
6.521	activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller Class Reference	2415
6.521.1	Detailed Description	2416
6.521.2	Constructor & Destructor Documentation	2416
6.521.2.1	MessageAckMarshaller	2416

6.521.2.2 ~MessageAckMarshaller	2416
6.521.3 Member Function Documentation	2416
6.521.3.1 createObject	2416
6.521.3.2 getDataStructureType	2416
6.521.3.3 looseMarshal	2417
6.521.3.4 looseUnmarshal	2417
6.521.3.5 tightMarshal1	2417
6.521.3.6 tightMarshal2	2418
6.521.3.7 tightUnmarshal	2418
6.522activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller Class Reference	2419
6.522.1 Detailed Description	2420
6.522.2 Constructor & Destructor Documentation	2420
6.522.2.1 MessageAckMarshaller	2420
6.522.2.2 ~MessageAckMarshaller	2420
6.522.3 Member Function Documentation	2420
6.522.3.1 createObject	2420
6.522.3.2 getDataStructureType	2420
6.522.3.3 looseMarshal	2421
6.522.3.4 looseUnmarshal	2421
6.522.3.5 tightMarshal1	2421
6.522.3.6 tightMarshal2	2422
6.522.3.7 tightUnmarshal	2422
6.523cms::MessageConsumer Class Reference	2423
6.523.1 Detailed Description	2423
6.523.2 Constructor & Destructor Documentation	2424
6.523.2.1 ~MessageConsumer	2424
6.523.3 Member Function Documentation	2424
6.523.3.1 getMessageListener	2424
6.523.3.2 getMessageSelector	2424
6.523.3.3 receive	2425
6.523.3.4 receive	2425
6.523.3.5 receiveNoWait	2425
6.523.3.6 setMessageListener	2425
6.524activemq::cmsutil::MessageCreator Class Reference	2426
6.524.1 Detailed Description	2426
6.524.2 Constructor & Destructor Documentation	2426

6.524.2.1	~MessageCreator	2426
6.524.3	Member Function Documentation	2426
6.524.3.1	createMessage	2426
6.525	activemq::commands::MessageDispatch Class Reference	2427
6.525.1	Constructor & Destructor Documentation	2428
6.525.1.1	MessageDispatch	2428
6.525.1.2	~MessageDispatch	2428
6.525.2	Member Function Documentation	2428
6.525.2.1	cloneDataStructure	2428
6.525.2.2	copyDataStructure	2428
6.525.2.3	equals	2428
6.525.2.4	getConsumerId	2429
6.525.2.5	getConsumerId	2429
6.525.2.6	getDataStructureType	2429
6.525.2.7	getDestination	2429
6.525.2.8	getDestination	2429
6.525.2.9	getMessage	2429
6.525.2.10	getMessage	2429
6.525.2.11	getRedeliveryCounter	2429
6.525.2.12	isMessageDispatch	2429
6.525.2.13	setConsumerId	2430
6.525.2.14	setDestination	2430
6.525.2.15	setMessage	2430
6.525.2.16	setRedeliveryCounter	2430
6.525.2.17	toString	2430
6.525.2.18	visit	2430
6.525.3	Field Documentation	2431
6.525.3.1	consumerId	2431
6.525.3.2	destination	2431
6.525.3.3	ID_MESSAGE_DISPATCH	2431
6.525.3.4	message	2431
6.525.3.5	redeliveryCounter	2431
6.526	activemq::core::MessageDispatchChannel Class Reference	2431
6.526.1	Constructor & Destructor Documentation	2433
6.526.1.1	MessageDispatchChannel	2433
6.526.1.2	~MessageDispatchChannel	2433

6.526.2 Member Function Documentation	2433
6.526.2.1 clear	2433
6.526.2.2 close	2433
6.526.2.3 dequeue	2433
6.526.2.4 dequeueNoWait	2433
6.526.2.5 enqueue	2434
6.526.2.6 enqueueFirst	2434
6.526.2.7 isClosed	2434
6.526.2.8 isEmpty	2434
6.526.2.9 isRunning	2434
6.526.2.10 lock	2434
6.526.2.11 notify	2435
6.526.2.12 notifyAll	2435
6.526.2.13 peek	2435
6.526.2.14 removeAll	2435
6.526.2.15 size	2436
6.526.2.16 start	2436
6.526.2.17 stop	2436
6.526.2.18 tryLock	2436
6.526.2.19 unlock	2436
6.526.2.20 wait	2436
6.526.2.21 lwait	2437
6.526.2.22 wait	2437
6.527 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller Class	
Reference	2438
6.527.1 Detailed Description	2439
6.527.2 Constructor & Destructor Documentation	2439
6.527.2.1 MessageDispatchMarshaller	2439
6.527.2.2 ~MessageDispatchMarshaller	2439
6.527.3 Member Function Documentation	2439
6.527.3.1 createObject	2439
6.527.3.2 getDataStructureType	2439
6.527.3.3 looseMarshal	2440
6.527.3.4 looseUnmarshal	2440
6.527.3.5 tightMarshal1	2440
6.527.3.6 tightMarshal2	2441
6.527.3.7 tightUnmarshal	2441

6.528	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	Class	
	Reference		2442
6.528.1	Detailed Description		2443
6.528.2	Constructor & Destructor Documentation		2443
	6.528.2.1 MessageDispatchMarshaller		2443
	6.528.2.2 ~MessageDispatchMarshaller		2443
6.528.3	Member Function Documentation		2443
	6.528.3.1 createObject		2443
	6.528.3.2 getDataStructureType		2443
	6.528.3.3 looseMarshal		2444
	6.528.3.4 looseUnmarshal		2444
	6.528.3.5 tightMarshal1		2444
	6.528.3.6 tightMarshal2		2445
	6.528.3.7 tightUnmarshal		2445
6.529	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	Class	
	Reference		2446
6.529.1	Detailed Description		2447
6.529.2	Constructor & Destructor Documentation		2447
	6.529.2.1 MessageDispatchMarshaller		2447
	6.529.2.2 ~MessageDispatchMarshaller		2447
6.529.3	Member Function Documentation		2447
	6.529.3.1 createObject		2447
	6.529.3.2 getDataStructureType		2447
	6.529.3.3 looseMarshal		2448
	6.529.3.4 looseUnmarshal		2448
	6.529.3.5 tightMarshal1		2448
	6.529.3.6 tightMarshal2		2449
	6.529.3.7 tightUnmarshal		2449
6.530	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	Class	
	Reference		2450
6.530.1	Detailed Description		2451
6.530.2	Constructor & Destructor Documentation		2451
	6.530.2.1 MessageDispatchMarshaller		2451
	6.530.2.2 ~MessageDispatchMarshaller		2451
6.530.3	Member Function Documentation		2451
	6.530.3.1 createObject		2451
	6.530.3.2 getDataStructureType		2451

6.530.3.3 looseMarshal	2452
6.530.3.4 looseUnmarshal	2452
6.530.3.5 tightMarshal1	2452
6.530.3.6 tightMarshal2	2453
6.530.3.7 tightUnmarshal	2453
6.531activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller Class	
Reference	2454
6.531.1 Detailed Description	2455
6.531.2 Constructor & Destructor Documentation	2455
6.531.2.1 MessageDispatchMarshaller	2455
6.531.2.2 ~MessageDispatchMarshaller	2455
6.531.3 Member Function Documentation	2455
6.531.3.1 createObject	2455
6.531.3.2 getDataStructureType	2455
6.531.3.3 looseMarshal	2456
6.531.3.4 looseUnmarshal	2456
6.531.3.5 tightMarshal1	2456
6.531.3.6 tightMarshal2	2457
6.531.3.7 tightUnmarshal	2457
6.532activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller Class	
Reference	2458
6.532.1 Detailed Description	2459
6.532.2 Constructor & Destructor Documentation	2459
6.532.2.1 MessageDispatchMarshaller	2459
6.532.2.2 ~MessageDispatchMarshaller	2459
6.532.3 Member Function Documentation	2459
6.532.3.1 createObject	2459
6.532.3.2 getDataStructureType	2459
6.532.3.3 looseMarshal	2460
6.532.3.4 looseUnmarshal	2460
6.532.3.5 tightMarshal1	2460
6.532.3.6 tightMarshal2	2461
6.532.3.7 tightUnmarshal	2461
6.533activemq::commands::MessageDispatchNotification Class Reference	2462
6.533.1 Constructor & Destructor Documentation	2463
6.533.1.1 MessageDispatchNotification	2463
6.533.1.2 ~MessageDispatchNotification	2463

6.533.2 Member Function Documentation	2463
6.533.2.1 cloneDataStructure	2463
6.533.2.2 copyDataStructure	2463
6.533.2.3 equals	2464
6.533.2.4 getConsumerId	2464
6.533.2.5 getConsumerId	2464
6.533.2.6 getDataStructureType	2464
6.533.2.7 getDeliverySequenceId	2465
6.533.2.8 getDestination	2465
6.533.2.9 getDestination	2465
6.533.2.10 getMessageId	2465
6.533.2.11 getMessageId	2465
6.533.2.12 sMessageDispatchNotification	2465
6.533.2.13 setConsumerId	2465
6.533.2.14 setDeliverySequenceId	2465
6.533.2.15 setDestination	2465
6.533.2.16 setMessageId	2465
6.533.2.17 toString	2465
6.533.2.18 visit	2466
6.533.3 Field Documentation	2466
6.533.3.1 consumerId	2466
6.533.3.2 deliverySequenceId	2466
6.533.3.3 destination	2466
6.533.3.4 ID_MESSAGE_DISPATCH_NOTIFICATION	2466
6.533.3.5 messageId	2466
6.534 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller	
Class Reference	2466
6.534.1 Detailed Description	2467
6.534.2 Constructor & Destructor Documentation	2468
6.534.2.1 MessageDispatchNotificationMarshaller	2468
6.534.2.2 ~MessageDispatchNotificationMarshaller	2468
6.534.3 Member Function Documentation	2468
6.534.3.1 createObject	2468
6.534.3.2 getDataStructureType	2468
6.534.3.3 looseMarshal	2468
6.534.3.4 looseUnmarshal	2469
6.534.3.5 tightMarshal1	2469

6.534.3.6 tightMarshal2	2469
6.534.3.7 tightUnmarshal	2470
6.535activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	
Class Reference	2470
6.535.1 Detailed Description	2471
6.535.2 Constructor & Destructor Documentation	2472
6.535.2.1 MessageDispatchNotificationMarshaller	2472
6.535.2.2 ~MessageDispatchNotificationMarshaller	2472
6.535.3 Member Function Documentation	2472
6.535.3.1 createObject	2472
6.535.3.2 getDataStructureType	2472
6.535.3.3 looseMarshal	2472
6.535.3.4 looseUnmarshal	2473
6.535.3.5 tightMarshal1	2473
6.535.3.6 tightMarshal2	2473
6.535.3.7 tightUnmarshal	2474
6.536activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	
Class Reference	2474
6.536.1 Detailed Description	2475
6.536.2 Constructor & Destructor Documentation	2476
6.536.2.1 MessageDispatchNotificationMarshaller	2476
6.536.2.2 ~MessageDispatchNotificationMarshaller	2476
6.536.3 Member Function Documentation	2476
6.536.3.1 createObject	2476
6.536.3.2 getDataStructureType	2476
6.536.3.3 looseMarshal	2476
6.536.3.4 looseUnmarshal	2477
6.536.3.5 tightMarshal1	2477
6.536.3.6 tightMarshal2	2477
6.536.3.7 tightUnmarshal	2478
6.537activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller	
Class Reference	2478
6.537.1 Detailed Description	2479
6.537.2 Constructor & Destructor Documentation	2480
6.537.2.1 MessageDispatchNotificationMarshaller	2480
6.537.2.2 ~MessageDispatchNotificationMarshaller	2480
6.537.3 Member Function Documentation	2480

6.537.3.1 createObject	2480
6.537.3.2 getDataStructureType	2480
6.537.3.3 looseMarshal	2480
6.537.3.4 looseUnmarshal	2481
6.537.3.5 tightMarshal1	2481
6.537.3.6 tightMarshal2	2481
6.537.3.7 tightUnmarshal	2482
6.538activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	
Class Reference	2482
6.538.1 Detailed Description	2483
6.538.2 Constructor & Destructor Documentation	2484
6.538.2.1 MessageDispatchNotificationMarshaller	2484
6.538.2.2 ~MessageDispatchNotificationMarshaller	2484
6.538.3 Member Function Documentation	2484
6.538.3.1 createObject	2484
6.538.3.2 getDataStructureType	2484
6.538.3.3 looseMarshal	2484
6.538.3.4 looseUnmarshal	2485
6.538.3.5 tightMarshal1	2485
6.538.3.6 tightMarshal2	2485
6.538.3.7 tightUnmarshal	2486
6.539activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	
Class Reference	2486
6.539.1 Detailed Description	2487
6.539.2 Constructor & Destructor Documentation	2488
6.539.2.1 MessageDispatchNotificationMarshaller	2488
6.539.2.2 ~MessageDispatchNotificationMarshaller	2488
6.539.3 Member Function Documentation	2488
6.539.3.1 createObject	2488
6.539.3.2 getDataStructureType	2488
6.539.3.3 looseMarshal	2488
6.539.3.4 looseUnmarshal	2489
6.539.3.5 tightMarshal1	2489
6.539.3.6 tightMarshal2	2489
6.539.3.7 tightUnmarshal	2490
6.540cms::MessageEnumeration Class Reference	2490
6.540.1 Detailed Description	2491

6.540.2 Constructor & Destructor Documentation	2491
6.540.2.1 ~MessageEnumeration	2491
6.540.3 Member Function Documentation	2491
6.540.3.1 hasMoreMessages	2491
6.540.3.2 nextMessage	2491
6.541cms::MessageEOFException Class Reference	2492
6.541.1 Detailed Description	2492
6.541.2 Constructor & Destructor Documentation	2493
6.541.2.1 MessageEOFException	2493
6.541.2.2 MessageEOFException	2493
6.541.2.3 MessageEOFException	2493
6.541.2.4 MessageEOFException	2493
6.541.2.5 ~MessageEOFException	2493
6.542cms::MessageFormatException Class Reference	2493
6.542.1 Detailed Description	2493
6.542.2 Constructor & Destructor Documentation	2494
6.542.2.1 MessageFormatException	2494
6.542.2.2 MessageFormatException	2494
6.542.2.3 MessageFormatException	2494
6.542.2.4 MessageFormatException	2494
6.542.2.5 ~MessageFormatException	2494
6.543activemq::commands::MessageId Class Reference	2494
6.543.1 Member Typedef Documentation	2496
6.543.1.1 COMPARATOR	2496
6.543.2 Constructor & Destructor Documentation	2496
6.543.2.1 MessageId	2496
6.543.2.2 MessageId	2496
6.543.2.3 MessageId	2496
6.543.2.4 MessageId	2496
6.543.2.5 MessageId	2496
6.543.2.6 MessageId	2496
6.543.2.7 ~MessageId	2496
6.543.3 Member Function Documentation	2496
6.543.3.1 cloneDataStructure	2496
6.543.3.2 compareTo	2496
6.543.3.3 copyDataStructure	2496

6.543.3.4 equals	2497
6.543.3.5 equals	2497
6.543.3.6 getBrokerSequenceId	2497
6.543.3.7 getDataStructureType	2497
6.543.3.8 getProducerId	2498
6.543.3.9 getProducerId	2498
6.543.3.10getProducerSequenceId	2498
6.543.3.11operator<	2498
6.543.3.12operator=	2498
6.543.3.13operator==	2498
6.543.3.14setBrokerSequenceId	2498
6.543.3.15setProducerId	2498
6.543.3.16setProducerSequenceId	2498
6.543.3.17set TextView	2498
6.543.3.18set Value	2498
6.543.3.19oString	2498
6.543.4 Field Documentation	2499
6.543.4.1 brokerSequenceId	2499
6.543.4.2 ID_MESSAGEID	2499
6.543.4.3 producerId	2499
6.543.4.4 producerSequenceId	2499
6.544activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class Reference	2499
6.544.1 Detailed Description	2500
6.544.2 Constructor & Destructor Documentation	2500
6.544.2.1 MessageIdMarshaller	2500
6.544.2.2 ~MessageIdMarshaller	2500
6.544.3 Member Function Documentation	2500
6.544.3.1 createObject	2500
6.544.3.2 getDataStructureType	2500
6.544.3.3 looseMarshal	2501
6.544.3.4 looseUnmarshal	2501
6.544.3.5 tightMarshal1	2501
6.544.3.6 tightMarshal2	2502
6.544.3.7 tightUnmarshal	2502
6.545activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class Reference	2503
6.545.1 Detailed Description	2504

6.545.2 Constructor & Destructor Documentation	2504
6.545.2.1 MessageIdMarshaller	2504
6.545.2.2 ~MessageIdMarshaller	2504
6.545.3 Member Function Documentation	2504
6.545.3.1 createObject	2504
6.545.3.2 getDataStructureType	2504
6.545.3.3 looseMarshal	2505
6.545.3.4 looseUnmarshal	2505
6.545.3.5 tightMarshal1	2505
6.545.3.6 tightMarshal2	2506
6.545.3.7 tightUnmarshal	2506
6.546activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class Reference	2507
6.546.1 Detailed Description	2508
6.546.2 Constructor & Destructor Documentation	2508
6.546.2.1 MessageIdMarshaller	2508
6.546.2.2 ~MessageIdMarshaller	2508
6.546.3 Member Function Documentation	2508
6.546.3.1 createObject	2508
6.546.3.2 getDataStructureType	2508
6.546.3.3 looseMarshal	2508
6.546.3.4 looseUnmarshal	2509
6.546.3.5 tightMarshal1	2509
6.546.3.6 tightMarshal2	2510
6.546.3.7 tightUnmarshal	2510
6.547activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class Reference	2510
6.547.1 Detailed Description	2511
6.547.2 Constructor & Destructor Documentation	2512
6.547.2.1 MessageIdMarshaller	2512
6.547.2.2 ~MessageIdMarshaller	2512
6.547.3 Member Function Documentation	2512
6.547.3.1 createObject	2512
6.547.3.2 getDataStructureType	2512
6.547.3.3 looseMarshal	2512
6.547.3.4 looseUnmarshal	2513
6.547.3.5 tightMarshal1	2513
6.547.3.6 tightMarshal2	2513

6.547.3.7 tightUnmarshal	2514
6.548activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller Class Reference	2514
6.548.1 Detailed Description	2515
6.548.2 Constructor & Destructor Documentation	2516
6.548.2.1 MessageIdMarshaller	2516
6.548.2.2 ~MessageIdMarshaller	2516
6.548.3 Member Function Documentation	2516
6.548.3.1 createObject	2516
6.548.3.2 getDataStructureType	2516
6.548.3.3 looseMarshal	2516
6.548.3.4 looseUnmarshal	2517
6.548.3.5 tightMarshal1	2517
6.548.3.6 tightMarshal2	2517
6.548.3.7 tightUnmarshal	2518
6.549activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller Class Reference	2518
6.549.1 Detailed Description	2519
6.549.2 Constructor & Destructor Documentation	2520
6.549.2.1 MessageIdMarshaller	2520
6.549.2.2 ~MessageIdMarshaller	2520
6.549.3 Member Function Documentation	2520
6.549.3.1 createObject	2520
6.549.3.2 getDataStructureType	2520
6.549.3.3 looseMarshal	2520
6.549.3.4 looseUnmarshal	2521
6.549.3.5 tightMarshal1	2521
6.549.3.6 tightMarshal2	2521
6.549.3.7 tightUnmarshal	2522
6.550cms::MessageListener Class Reference	2522
6.550.1 Detailed Description	2523
6.550.2 Constructor & Destructor Documentation	2523
6.550.2.1 ~MessageListener	2523
6.550.3 Member Function Documentation	2523
6.550.3.1 onMessage	2523
6.551activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference	2523
6.551.1 Detailed Description	2524
6.551.2 Constructor & Destructor Documentation	2524

6.551.2.1 MessageMarshaller	2524
6.551.2.2 ~MessageMarshaller	2524
6.551.3 Member Function Documentation	2524
6.551.3.1 looseMarshal	2524
6.551.3.2 looseUnmarshal	2525
6.551.3.3 tightMarshal1	2526
6.551.3.4 tightMarshal2	2526
6.551.3.5 tightUnmarshal	2527
6.552activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class Reference	2527
6.552.1 Detailed Description	2528
6.552.2 Constructor & Destructor Documentation	2529
6.552.2.1 MessageMarshaller	2529
6.552.2.2 ~MessageMarshaller	2529
6.552.3 Member Function Documentation	2529
6.552.3.1 looseMarshal	2529
6.552.3.2 looseUnmarshal	2529
6.552.3.3 tightMarshal1	2530
6.552.3.4 tightMarshal2	2531
6.552.3.5 tightUnmarshal	2531
6.553activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class Reference	2532
6.553.1 Detailed Description	2533
6.553.2 Constructor & Destructor Documentation	2533
6.553.2.1 MessageMarshaller	2533
6.553.2.2 ~MessageMarshaller	2533
6.553.3 Member Function Documentation	2533
6.553.3.1 looseMarshal	2533
6.553.3.2 looseUnmarshal	2533
6.553.3.3 tightMarshal1	2534
6.553.3.4 tightMarshal2	2535
6.553.3.5 tightUnmarshal	2535
6.554activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class Reference	2536
6.554.1 Detailed Description	2537
6.554.2 Constructor & Destructor Documentation	2537
6.554.2.1 MessageMarshaller	2537
6.554.2.2 ~MessageMarshaller	2537
6.554.3 Member Function Documentation	2537

6.554.3.1 looseMarshal	2537
6.554.3.2 looseUnmarshal	2537
6.554.3.3 tightMarshal1	2538
6.554.3.4 tightMarshal2	2539
6.554.3.5 tightUnmarshal	2539
6.555activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference	2540
6.555.1 Detailed Description	2541
6.555.2 Constructor & Destructor Documentation	2541
6.555.2.1 MessageMarshaller	2541
6.555.2.2 ~MessageMarshaller	2541
6.555.3 Member Function Documentation	2541
6.555.3.1 looseMarshal	2541
6.555.3.2 looseUnmarshal	2541
6.555.3.3 tightMarshal1	2542
6.555.3.4 tightMarshal2	2543
6.555.3.5 tightUnmarshal	2543
6.556activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class Reference	2544
6.556.1 Detailed Description	2545
6.556.2 Constructor & Destructor Documentation	2545
6.556.2.1 MessageMarshaller	2545
6.556.2.2 ~MessageMarshaller	2545
6.556.3 Member Function Documentation	2545
6.556.3.1 looseMarshal	2545
6.556.3.2 looseUnmarshal	2545
6.556.3.3 tightMarshal1	2546
6.556.3.4 tightMarshal2	2547
6.556.3.5 tightUnmarshal	2547
6.557cms::MessageNotReadableException Class Reference	2548
6.557.1 Detailed Description	2548
6.557.2 Constructor & Destructor Documentation	2549
6.557.2.1 MessageNotReadableException	2549
6.557.2.2 MessageNotReadableException	2549
6.557.2.3 MessageNotReadableException	2549
6.557.2.4 MessageNotReadableException	2549
6.557.2.5 ~MessageNotReadableException	2549
6.558cms::MessageNotWriteableException Class Reference	2549

6.558.1 Detailed Description	2549
6.558.2 Constructor & Destructor Documentation	2550
6.558.2.1 MessageNotWriteableException	2550
6.558.2.2 MessageNotWriteableException	2550
6.558.2.3 MessageNotWriteableException	2550
6.558.2.4 MessageNotWriteableException	2550
6.558.2.5 ~MessageNotWriteableException	2550
6.559cms::MessageProducer Class Reference	2550
6.559.1 Detailed Description	2551
6.559.2 Constructor & Destructor Documentation	2552
6.559.2.1 ~MessageProducer	2552
6.559.3 Member Function Documentation	2552
6.559.3.1 getDeliveryMode	2552
6.559.3.2 getDisableMessageID	2552
6.559.3.3 getDisableMessageTimeStamp	2553
6.559.3.4 getPriority	2553
6.559.3.5 getTimeToLive	2553
6.559.3.6 send	2553
6.559.3.7 send	2554
6.559.3.8 send	2555
6.559.3.9 send	2555
6.559.3.10 setDeliveryMode	2556
6.559.3.11 setDisableMessageID	2556
6.559.3.12 setDisableMessageTimeStamp	2556
6.559.3.13 setPriority	2556
6.559.3.14 setTimeToLive	2557
6.560activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference	2557
6.560.1 Detailed Description	2559
6.560.2 Constructor & Destructor Documentation	2559
6.560.2.1 MessagePropertyInterceptor	2559
6.560.2.2 ~MessagePropertyInterceptor	2559
6.560.3 Member Function Documentation	2559
6.560.3.1 getBooleanProperty	2559
6.560.3.2 getByteProperty	2559
6.560.3.3 getDoubleProperty	2560
6.560.3.4 getFloatProperty	2560

6.560.3.5	getIntProperty	2560
6.560.3.6	getLongProperty	2561
6.560.3.7	getShortProperty	2561
6.560.3.8	getStringProperty	2561
6.560.3.9	setBooleanProperty	2561
6.560.3.10	setByteProperty	2562
6.560.3.11	setDoubleProperty	2562
6.560.3.12	setFloatProperty	2562
6.560.3.13	setIntProperty	2562
6.560.3.14	setLongProperty	2562
6.560.3.15	setShortProperty	2563
6.560.3.16	setStringProperty	2563
6.561	activemq::commands::MessagePull Class Reference	2563
6.561.1	Constructor & Destructor Documentation	2565
6.561.1.1	MessagePull	2565
6.561.1.2	~MessagePull	2565
6.561.2	Member Function Documentation	2565
6.561.2.1	cloneDataStructure	2565
6.561.2.2	copyDataStructure	2565
6.561.2.3	equals	2565
6.561.2.4	getConsumerId	2566
6.561.2.5	getConsumerId	2566
6.561.2.6	getCorrelationId	2566
6.561.2.7	getCorrelationId	2566
6.561.2.8	getDataStructureType	2566
6.561.2.9	getDestination	2567
6.561.2.10	getDestination	2567
6.561.2.11	getMessageId	2567
6.561.2.12	getMessageId	2567
6.561.2.13	getTimeout	2567
6.561.2.14	setConsumerId	2567
6.561.2.15	setCorrelationId	2567
6.561.2.16	setDestination	2567
6.561.2.17	setMessageId	2567
6.561.2.18	setTimeout	2567
6.561.2.19	toString	2567

6.561.2.20	visit	2567
6.561.3	Field Documentation	2568
6.561.3.1	consumerId	2568
6.561.3.2	correlationId	2568
6.561.3.3	destination	2568
6.561.3.4	ID_MESSAGEPULL	2568
6.561.3.5	messageId	2568
6.561.3.6	timeout	2568
6.562	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller Class Reference	2568
6.562.1	Detailed Description	2569
6.562.2	Constructor & Destructor Documentation	2569
6.562.2.1	MessagePullMarshaller	2569
6.562.2.2	~MessagePullMarshaller	2569
6.562.3	Member Function Documentation	2569
6.562.3.1	createObject	2569
6.562.3.2	getDataStructureType	2570
6.562.3.3	looseMarshal	2570
6.562.3.4	looseUnmarshal	2570
6.562.3.5	tightMarshal1	2571
6.562.3.6	tightMarshal2	2571
6.562.3.7	tightUnmarshal	2571
6.563	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller Class Reference	2572
6.563.1	Detailed Description	2573
6.563.2	Constructor & Destructor Documentation	2573
6.563.2.1	MessagePullMarshaller	2573
6.563.2.2	~MessagePullMarshaller	2573
6.563.3	Member Function Documentation	2573
6.563.3.1	createObject	2573
6.563.3.2	getDataStructureType	2573
6.563.3.3	looseMarshal	2574
6.563.3.4	looseUnmarshal	2574
6.563.3.5	tightMarshal1	2574
6.563.3.6	tightMarshal2	2575
6.563.3.7	tightUnmarshal	2575
6.564	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller Class Reference	2576
6.564.1	Detailed Description	2577

6.564.2 Constructor & Destructor Documentation	2577
6.564.2.1 MessagePullMarshaller	2577
6.564.2.2 ~MessagePullMarshaller	2577
6.564.3 Member Function Documentation	2577
6.564.3.1 createObject	2577
6.564.3.2 getDataStructureType	2577
6.564.3.3 looseMarshal	2578
6.564.3.4 looseUnmarshal	2578
6.564.3.5 tightMarshal1	2578
6.564.3.6 tightMarshal2	2579
6.564.3.7 tightUnmarshal	2579
6.565activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller Class Reference	2580
6.565.1 Detailed Description	2581
6.565.2 Constructor & Destructor Documentation	2581
6.565.2.1 MessagePullMarshaller	2581
6.565.2.2 ~MessagePullMarshaller	2581
6.565.3 Member Function Documentation	2581
6.565.3.1 createObject	2581
6.565.3.2 getDataStructureType	2581
6.565.3.3 looseMarshal	2582
6.565.3.4 looseUnmarshal	2582
6.565.3.5 tightMarshal1	2582
6.565.3.6 tightMarshal2	2583
6.565.3.7 tightUnmarshal	2583
6.566activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller Class Reference	2584
6.566.1 Detailed Description	2585
6.566.2 Constructor & Destructor Documentation	2585
6.566.2.1 MessagePullMarshaller	2585
6.566.2.2 ~MessagePullMarshaller	2585
6.566.3 Member Function Documentation	2585
6.566.3.1 createObject	2585
6.566.3.2 getDataStructureType	2585
6.566.3.3 looseMarshal	2586
6.566.3.4 looseUnmarshal	2586
6.566.3.5 tightMarshal1	2586
6.566.3.6 tightMarshal2	2587

6.566.3.7 tightUnmarshal	2587
6.567activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller Class Reference	2588
6.567.1 Detailed Description	2589
6.567.2 Constructor & Destructor Documentation	2589
6.567.2.1 MessagePullMarshaller	2589
6.567.2.2 ~MessagePullMarshaller	2589
6.567.3 Member Function Documentation	2589
6.567.3.1 createObject	2589
6.567.3.2 getDataStructureType	2589
6.567.3.3 looseMarshal	2590
6.567.3.4 looseUnmarshal	2590
6.567.3.5 tightMarshal1	2590
6.567.3.6 tightMarshal2	2591
6.567.3.7 tightUnmarshal	2591
6.568activemq::transport::mock::MockTransport Class Reference	2592
6.568.1 Detailed Description	2594
6.568.2 Constructor & Destructor Documentation	2594
6.568.2.1 MockTransport	2594
6.568.2.2 ~MockTransport	2594
6.568.3 Member Function Documentation	2594
6.568.3.1 close	2594
6.568.3.2 fireCommand	2595
6.568.3.3 fireException	2595
6.568.3.4 getInstance	2596
6.568.3.5 getNumReceivedMessageBeforeFail	2596
6.568.3.6 getNumReceivedMessages	2596
6.568.3.7 getNumSentKeepAlives	2596
6.568.3.8 getNumSentKeepAlivesBeforeFail	2596
6.568.3.9 getNumSentMessageBeforeFail	2596
6.568.3.10getNumSentMessages	2596
6.568.3.11getRemoteAddress	2596
6.568.3.12getTransportListener	2596
6.568.3.13getWireFormat	2596
6.568.3.14sClosed	2597
6.568.3.15sConnected	2597
6.568.3.16sFailOnClose	2597

6.568.3.17sFailOnKeepAliveSends	2597
6.568.3.18sFailOnReceiveMessage	2597
6.568.3.19sFailOnSendMessage	2597
6.568.3.20sFailOnStart	2597
6.568.3.21sFailOnStop	2597
6.568.3.22sFault Tolerant	2597
6.568.3.23narrow	2598
6.568.3.24neway	2598
6.568.3.25reconnect	2598
6.568.3.26request	2599
6.568.3.27request	2599
6.568.3.28set FailOnClose	2600
6.568.3.29set FailOnKeepAliveSends	2600
6.568.3.30set FailOnReceiveMessage	2600
6.568.3.31set FailOnSendMessage	2600
6.568.3.32set FailOnStart	2600
6.568.3.33set FailOnStop	2600
6.568.3.34set NumReceivedMessageBeforeFail	2600
6.568.3.35set NumReceivedMessages	2600
6.568.3.36set NumSentKeepAlives	2600
6.568.3.37set NumSentKeepAlivesBeforeFail	2600
6.568.3.38set NumSentMessageBeforeFail	2600
6.568.3.39set NumSentMessages	2600
6.568.3.40set OutgoingListener	2600
6.568.3.41set ResponseBuilder	2601
6.568.3.42set TransportListener	2601
6.568.3.43set WireFormat	2601
6.568.3.44start	2601
6.568.3.45stop	2601
6.569activemq::transport::mock::MockTransportFactory Class Reference	2602
6.569.1 Detailed Description	2602
6.569.2 Constructor & Destructor Documentation	2603
6.569.2.1 ~MockTransportFactory	2603
6.569.3 Member Function Documentation	2603
6.569.3.1 create	2603
6.569.3.2 createComposite	2603

6.569.3.3 doCreateComposite	2603
6.570decaf::util::concurrent::Mutex Class Reference	2604
6.570.1 Detailed Description	2605
6.570.2 Constructor & Destructor Documentation	2605
6.570.2.1 Mutex	2605
6.570.2.2 ~Mutex	2605
6.570.3 Member Function Documentation	2605
6.570.3.1 lock	2605
6.570.3.2 notify	2606
6.570.3.3 notifyAll	2606
6.570.3.4 tryLock	2606
6.570.3.5 unlock	2607
6.570.3.6 wait	2607
6.570.3.7 wait	2608
6.570.3.8 wait	2608
6.571decaf::util::concurrent::MutexHandle Class Reference	2609
6.571.1 Constructor & Destructor Documentation	2609
6.571.1.1 MutexHandle	2609
6.571.1.2 ~MutexHandle	2609
6.571.1.3 MutexHandle	2609
6.571.1.4 ~MutexHandle	2609
6.571.2 Field Documentation	2609
6.571.2.1 lock_count	2609
6.571.2.2 lock_owner	2609
6.571.2.3 mutex	2609
6.571.2.4 mutex	2609
6.572decaf::internal::util::concurrent::MutexImpl Class Reference	2609
6.572.1 Member Function Documentation	2610
6.572.1.1 create	2610
6.572.1.2 destroy	2610
6.572.1.3 lock	2610
6.572.1.4 trylock	2611
6.572.1.5 unlock	2611
6.573decaf::internal::net::Network Class Reference	2611
6.573.1 Detailed Description	2612
6.573.2 Constructor & Destructor Documentation	2612

6.573.2.1 Network	2612
6.573.2.2 ~Network	2612
6.573.3 Member Function Documentation	2612
6.573.3.1 addAsResource	2612
6.573.3.2 addNetworkResource	2612
6.573.3.3 getNetworkRuntime	2613
6.573.3.4 getRuntimeLock	2613
6.573.3.5 initializeNetworking	2613
6.573.3.6 shutdownNetworking	2613
6.574activemq::commands::NetworkBridgeFilter Class Reference	2613
6.574.1 Constructor & Destructor Documentation	2615
6.574.1.1 NetworkBridgeFilter	2615
6.574.1.2 ~NetworkBridgeFilter	2615
6.574.2 Member Function Documentation	2615
6.574.2.1 cloneDataStructure	2615
6.574.2.2 copyDataStructure	2615
6.574.2.3 equals	2615
6.574.2.4 getDataStructureType	2615
6.574.2.5 getNetworkBrokerId	2616
6.574.2.6 getNetworkBrokerId	2616
6.574.2.7 getNetworkTTL	2616
6.574.2.8 setNetworkBrokerId	2616
6.574.2.9 setNetworkTTL	2616
6.574.2.10toString	2616
6.574.3 Field Documentation	2616
6.574.3.1 ID_NETWORKBRIDGEFILTER	2616
6.574.3.2 networkBrokerId	2616
6.574.3.3 networkTTL	2616
6.575activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller Class Reference	2617
6.575.1 Detailed Description	2618
6.575.2 Constructor & Destructor Documentation	2618
6.575.2.1 NetworkBridgeFilterMarshaller	2618
6.575.2.2 ~NetworkBridgeFilterMarshaller	2618
6.575.3 Member Function Documentation	2618
6.575.3.1 createObject	2618
6.575.3.2 getDataStructureType	2618

6.575.3.3 looseMarshal	2618
6.575.3.4 looseUnmarshal	2619
6.575.3.5 tightMarshal1	2619
6.575.3.6 tightMarshal2	2620
6.575.3.7 tightUnmarshal	2620
6.576activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller Class	
Reference	2620
6.576.1 Detailed Description	2621
6.576.2 Constructor & Destructor Documentation	2622
6.576.2.1 NetworkBridgeFilterMarshaller	2622
6.576.2.2 ~NetworkBridgeFilterMarshaller	2622
6.576.3 Member Function Documentation	2622
6.576.3.1 createObject	2622
6.576.3.2 getDataStructureType	2622
6.576.3.3 looseMarshal	2622
6.576.3.4 looseUnmarshal	2623
6.576.3.5 tightMarshal1	2623
6.576.3.6 tightMarshal2	2623
6.576.3.7 tightUnmarshal	2624
6.577activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller Class	
Reference	2624
6.577.1 Detailed Description	2625
6.577.2 Constructor & Destructor Documentation	2626
6.577.2.1 NetworkBridgeFilterMarshaller	2626
6.577.2.2 ~NetworkBridgeFilterMarshaller	2626
6.577.3 Member Function Documentation	2626
6.577.3.1 createObject	2626
6.577.3.2 getDataStructureType	2626
6.577.3.3 looseMarshal	2626
6.577.3.4 looseUnmarshal	2627
6.577.3.5 tightMarshal1	2627
6.577.3.6 tightMarshal2	2627
6.577.3.7 tightUnmarshal	2628
6.578activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller Class	
Reference	2628
6.578.1 Detailed Description	2629
6.578.2 Constructor & Destructor Documentation	2630

6.578.2.1 NetworkBridgeFilterMarshaller	2630
6.578.2.2 ~NetworkBridgeFilterMarshaller	2630
6.578.3 Member Function Documentation	2630
6.578.3.1 createObject	2630
6.578.3.2 getDataStructureType	2630
6.578.3.3 looseMarshal	2630
6.578.3.4 looseUnmarshal	2631
6.578.3.5 tightMarshal1	2631
6.578.3.6 tightMarshal2	2631
6.578.3.7 tightUnmarshal	2632
6.579activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller Class	
Reference	2632
6.579.1 Detailed Description	2633
6.579.2 Constructor & Destructor Documentation	2634
6.579.2.1 NetworkBridgeFilterMarshaller	2634
6.579.2.2 ~NetworkBridgeFilterMarshaller	2634
6.579.3 Member Function Documentation	2634
6.579.3.1 createObject	2634
6.579.3.2 getDataStructureType	2634
6.579.3.3 looseMarshal	2634
6.579.3.4 looseUnmarshal	2635
6.579.3.5 tightMarshal1	2635
6.579.3.6 tightMarshal2	2635
6.579.3.7 tightUnmarshal	2636
6.580activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller Class	
Reference	2636
6.580.1 Detailed Description	2637
6.580.2 Constructor & Destructor Documentation	2638
6.580.2.1 NetworkBridgeFilterMarshaller	2638
6.580.2.2 ~NetworkBridgeFilterMarshaller	2638
6.580.3 Member Function Documentation	2638
6.580.3.1 createObject	2638
6.580.3.2 getDataStructureType	2638
6.580.3.3 looseMarshal	2638
6.580.3.4 looseUnmarshal	2639
6.580.3.5 tightMarshal1	2639
6.580.3.6 tightMarshal2	2639

6.580.3.7 tightUnmarshal	2640
6.581decaf::net::NoRouteToHostException Class Reference	2640
6.581.1 Constructor & Destructor Documentation	2641
6.581.1.1 NoRouteToHostException	2641
6.581.1.2 NoRouteToHostException	2641
6.581.1.3 NoRouteToHostException	2641
6.581.1.4 NoRouteToHostException	2642
6.581.1.5 NoRouteToHostException	2642
6.581.1.6 NoRouteToHostException	2642
6.581.1.7 ~NoRouteToHostException	2642
6.581.2 Member Function Documentation	2642
6.581.2.1 clone	2642
6.582decaf::security::NoSuchAlgorithmException Class Reference	2643
6.582.1 Constructor & Destructor Documentation	2644
6.582.1.1 NoSuchAlgorithmException	2644
6.582.1.2 NoSuchAlgorithmException	2644
6.582.1.3 NoSuchAlgorithmException	2644
6.582.1.4 NoSuchAlgorithmException	2644
6.582.1.5 NoSuchAlgorithmException	2644
6.582.1.6 NoSuchAlgorithmException	2645
6.582.1.7 ~NoSuchAlgorithmException	2645
6.582.2 Member Function Documentation	2645
6.582.2.1 clone	2645
6.583decaf::lang::exceptions::NoSuchElementException Class Reference	2645
6.583.1 Constructor & Destructor Documentation	2646
6.583.1.1 NoSuchElementException	2646
6.583.1.2 NoSuchElementException	2646
6.583.1.3 NoSuchElementException	2646
6.583.1.4 NoSuchElementException	2647
6.583.1.5 NoSuchElementException	2647
6.583.1.6 NoSuchElementException	2647
6.583.1.7 ~NoSuchElementException	2647
6.583.2 Member Function Documentation	2647
6.583.2.1 clone	2647
6.584decaf::security::NoSuchProviderException Class Reference	2648
6.584.1 Constructor & Destructor Documentation	2649

6.584.1.1 NoSuchProviderException	2649
6.584.1.2 NoSuchProviderException	2649
6.584.1.3 NoSuchProviderException	2649
6.584.1.4 NoSuchProviderException	2649
6.584.1.5 NoSuchProviderException	2649
6.584.1.6 NoSuchProviderException	2650
6.584.1.7 ~NoSuchProviderException	2650
6.584.2 Member Function Documentation	2650
6.584.2.1 clone	2650
6.585decaf::lang::exceptions::NullPointerException Class Reference	2650
6.585.1 Constructor & Destructor Documentation	2651
6.585.1.1 NullPointerException	2651
6.585.1.2 NullPointerException	2651
6.585.1.3 NullPointerException	2651
6.585.1.4 NullPointerException	2652
6.585.1.5 NullPointerException	2652
6.585.1.6 NullPointerException	2652
6.585.1.7 ~NullPointerException	2652
6.585.2 Member Function Documentation	2652
6.585.2.1 clone	2652
6.586decaf::lang::Number Class Reference	2653
6.586.1 Detailed Description	2653
6.586.2 Constructor & Destructor Documentation	2654
6.586.2.1 ~Number	2654
6.586.3 Member Function Documentation	2654
6.586.3.1 byteValue	2654
6.586.3.2 doubleValue	2654
6.586.3.3 floatValue	2654
6.586.3.4 intValue	2654
6.586.3.5 longValue	2654
6.586.3.6 shortValue	2655
6.587decaf::lang::exceptions::NumberFormatException Class Reference	2655
6.587.1 Constructor & Destructor Documentation	2656
6.587.1.1 NumberFormatException	2656
6.587.1.2 NumberFormatException	2656
6.587.1.3 NumberFormatException	2656

6.587.1.4 NumberFormatException	2656
6.587.1.5 NumberFormatException	2657
6.587.1.6 NumberFormatException	2657
6.587.1.7 ~NumberFormatException	2657
6.587.2 Member Function Documentation	2657
6.587.2.1 clone	2657
6.588cms::ObjectMessage Class Reference	2658
6.588.1 Detailed Description	2658
6.588.2 Constructor & Destructor Documentation	2658
6.588.2.1 ~ObjectMessage	2658
6.589decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference	2658
6.589.1 Detailed Description	2659
6.589.2 Constructor & Destructor Documentation	2660
6.589.2.1 OpenSSLContextSpi	2660
6.589.2.2 ~OpenSSLContextSpi	2660
6.589.3 Member Function Documentation	2660
6.589.3.1 providerGetServerSocketFactory	2660
6.589.3.2 providerGetSocketFactory	2660
6.589.3.3 providerInit	2661
6.589.4 Friends And Related Function Documentation	2661
6.589.4.1 OpenSSLSocket	2661
6.589.4.2 OpenSSLSocketFactory	2661
6.590decaf::internal::net::ssl::openssl::OpenSSLParameters Class Reference	2661
6.590.1 Detailed Description	2662
6.590.2 Constructor & Destructor Documentation	2662
6.590.2.1 ~OpenSSLParameters	2662
6.590.3 Member Function Documentation	2662
6.590.3.1 clone	2662
6.590.3.2 getEnabledCipherSuites	2663
6.590.3.3 getEnabledProtocols	2663
6.590.3.4 getNeedClientAuth	2663
6.590.3.5 getSupportedCipherSuites	2663
6.590.3.6 getSupportedProtocols	2663
6.590.3.7 getUseClientMode	2663
6.590.3.8 getWantClientAuth	2663
6.590.3.9 setEnabledCipherSuites	2663

6.590.3.10	setEnabledProtocols	2663
6.590.3.11	setNeedClientAuth	2663
6.590.3.12	setUseClientMode	2663
6.590.3.13	setWantClientAuth	2663
6.591	decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference	2664
6.591.1	Detailed Description	2665
6.591.2	Constructor & Destructor Documentation	2666
6.591.2.1	OpenSSLServerSocket	2666
6.591.2.2	~OpenSSLServerSocket	2666
6.591.3	Member Function Documentation	2666
6.591.3.1	accept	2666
6.591.3.2	setEnabledCipherSuites	2666
6.591.3.3	setEnabledProtocols	2667
6.591.3.4	getNeedClientAuth	2667
6.591.3.5	getSupportedCipherSuites	2667
6.591.3.6	getSupportedProtocols	2667
6.591.3.7	getWantClientAuth	2668
6.591.3.8	setEnabledCipherSuites	2668
6.591.3.9	setEnabledProtocols	2668
6.591.3.10	setNeedClientAuth	2668
6.591.3.11	setWantClientAuth	2669
6.592	decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference	2669
6.592.1	Detailed Description	2671
6.592.2	Constructor & Destructor Documentation	2671
6.592.2.1	OpenSSLServerSocketFactory	2671
6.592.2.2	~OpenSSLServerSocketFactory	2671
6.592.3	Member Function Documentation	2671
6.592.3.1	createServerSocket	2671
6.592.3.2	createServerSocket	2671
6.592.3.3	createServerSocket	2672
6.592.3.4	createServerSocket	2672
6.592.3.5	getDefaultCipherSuites	2673
6.592.3.6	getSupportedCipherSuites	2673
6.593	decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference	2673
6.593.1	Detailed Description	2678
6.593.2	Constructor & Destructor Documentation	2678

6.593.2.1 OpenSSLSocket	2678
6.593.2.2 OpenSSLSocket	2678
6.593.2.3 OpenSSLSocket	2678
6.593.2.4 OpenSSLSocket	2678
6.593.2.5 OpenSSLSocket	2678
6.593.2.6 ~OpenSSLSocket	2678
6.593.3 Member Function Documentation	2678
6.593.3.1 available	2678
6.593.3.2 close	2679
6.593.3.3 connect	2679
6.593.3.4 getEnabledCipherSuites	2679
6.593.3.5 getEnabledProtocols	2680
6.593.3.6 getInputStream	2680
6.593.3.7 getNeedClientAuth	2680
6.593.3.8 getOutputStream	2681
6.593.3.9 getSupportedCipherSuites	2681
6.593.3.10getSupportedProtocols	2681
6.593.3.11getUseClientMode	2681
6.593.3.12getWantClientAuth	2682
6.593.3.13read	2682
6.593.3.14endUrgentData	2682
6.593.3.15setEnabledCipherSuites	2683
6.593.3.16setEnabledProtocols	2683
6.593.3.17setNeedClientAuth	2683
6.593.3.18setOOBInline	2684
6.593.3.19setUseClientMode	2684
6.593.3.20setWantClientAuth	2684
6.593.3.21shutdownInput	2685
6.593.3.22shutdownOutput	2685
6.593.3.23startHandshake	2685
6.593.3.24write	2685
6.594decaf::internal::net::ssl::openssl::OpenSSLSocketException Class Reference	2686
6.594.1 Detailed Description	2687
6.594.2 Constructor & Destructor Documentation	2687
6.594.2.1 OpenSSLSocketException	2687
6.594.2.2 OpenSSLSocketException	2687

6.594.2.3 OpenSSLSocketException	2687
6.594.2.4 OpenSSLSocketException	2688
6.594.2.5 OpenSSLSocketException	2688
6.594.2.6 OpenSSLSocketException	2688
6.594.2.7 OpenSSLSocketException	2688
6.594.2.8 ~OpenSSLSocketException	2689
6.594.3 Member Function Documentation	2689
6.594.3.1 clone	2689
6.594.3.2 getErrorString	2689
6.595decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference	2689
6.595.1 Detailed Description	2692
6.595.2 Constructor & Destructor Documentation	2692
6.595.2.1 OpenSSLSocketFactory	2692
6.595.2.2 ~OpenSSLSocketFactory	2692
6.595.3 Member Function Documentation	2692
6.595.3.1 createSocket	2692
6.595.3.2 createSocket	2692
6.595.3.3 createSocket	2693
6.595.3.4 createSocket	2693
6.595.3.5 createSocket	2694
6.595.3.6 createSocket	2694
6.595.3.7 getDefaultCipherSuites	2695
6.595.3.8 getSupportedCipherSuites	2695
6.596decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream Class Reference	2696
6.596.1 Detailed Description	2696
6.596.2 Constructor & Destructor Documentation	2697
6.596.2.1 OpenSSLSocketInputStream	2697
6.596.2.2 ~OpenSSLSocketInputStream	2697
6.596.3 Member Function Documentation	2697
6.596.3.1 available	2697
6.596.3.2 close	2697
6.596.3.3 doReadArrayBounded	2697
6.596.3.4 doReadByte	2698
6.596.3.5 skip	2698
6.597decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream Class Reference	2698
6.597.1 Detailed Description	2699

6.597.2 Constructor & Destructor Documentation	2699
6.597.2.1 OpenSSLSocketOutputStream	2699
6.597.2.2 ~OpenSSLSocketOutputStream	2699
6.597.3 Member Function Documentation	2699
6.597.3.1 close	2699
6.597.3.2 doWriteArrayBounded	2699
6.597.3.3 doWriteByte	2700
6.598activemq::wireformat::openwire::OpenWireFormat Class Reference	2700
6.598.1 Constructor & Destructor Documentation	2703
6.598.1.1 OpenWireFormat	2703
6.598.1.2 ~OpenWireFormat	2703
6.598.2 Member Function Documentation	2703
6.598.2.1 addMarshaller	2703
6.598.2.2 createNegotiator	2703
6.598.2.3 destroyMarshalers	2704
6.598.2.4 doUnmarshal	2704
6.598.2.5 getCacheSize	2704
6.598.2.6 getMaxInactivityDuration	2704
6.598.2.7 getMaxInactivityDurationInitialDelay	2705
6.598.2.8 getPreferredWireFormatInfo	2705
6.598.2.9 getVersion	2705
6.598.2.10hasNegotiator	2705
6.598.2.11inReceive	2705
6.598.2.12sCacheEnabled	2706
6.598.2.13sSizePrefixDisabled	2706
6.598.2.14sStackTraceEnabled	2706
6.598.2.15sTcpNoDelayEnabled	2706
6.598.2.16sTightEncodingEnabled	2706
6.598.2.17ooseMarshalNestedObject	2707
6.598.2.18ooseUnmarshalNestedObject	2707
6.598.2.19marshal	2707
6.598.2.20renegotiateWireFormat	2708
6.598.2.21setCacheEnabled	2708
6.598.2.22setCacheSize	2708
6.598.2.23setMaxInactivityDuration	2708
6.598.2.24setMaxInactivityDurationInitialDelay	2709

6.598.2.25	setPreferedWireFormatInfo	2709
6.598.2.26	setSizePrefixDisabled	2709
6.598.2.27	setStackTraceEnabled	2709
6.598.2.28	setTcpNoDelayEnabled	2709
6.598.2.29	setTightEncodingEnabled	2710
6.598.2.30	setVersion	2710
6.598.2.31	tightMarshalNestedObject1	2710
6.598.2.32	tightMarshalNestedObject2	2710
6.598.2.33	tightUnmarshalNestedObject	2711
6.598.2.34	unmarshal	2711
6.598.3	Field Documentation	2712
6.598.3.1	DEFAULT_VERSION	2712
6.598.3.2	NULL_TYPE	2712
6.599	activemq::wireformat::openwire::OpenWireFormatFactory Class Reference	2712
6.599.1	Constructor & Destructor Documentation	2712
6.599.1.1	OpenWireFormatFactory	2712
6.599.1.2	~OpenWireFormatFactory	2713
6.599.2	Member Function Documentation	2713
6.599.2.1	createWireFormat	2713
6.600	activemq::wireformat::openwire::OpenWireFormatNegotiator Class Reference	2713
6.600.1	Constructor & Destructor Documentation	2714
6.600.1.1	OpenWireFormatNegotiator	2714
6.600.1.2	~OpenWireFormatNegotiator	2714
6.600.2	Member Function Documentation	2714
6.600.2.1	close	2714
6.600.2.2	onCommand	2715
6.600.2.3	oneway	2715
6.600.2.4	onTransportException	2715
6.600.2.5	request	2716
6.600.2.6	request	2716
6.600.2.7	start	2716
6.601	activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference	2717
6.601.1	Constructor & Destructor Documentation	2718
6.601.1.1	OpenWireResponseBuilder	2718
6.601.1.2	~OpenWireResponseBuilder	2718
6.601.2	Member Function Documentation	2718

6.601.2.1 buildIncomingCommands	2718
6.601.2.2 buildResponse	2718
6.602decaf::io::OutputStream Class Reference	2718
6.602.1 Detailed Description	2720
6.602.2 Constructor & Destructor Documentation	2720
6.602.2.1 OutputStream	2720
6.602.2.2 ~OutputStream	2720
6.602.3 Member Function Documentation	2720
6.602.3.1 close	2720
6.602.3.2 doWriteArray	2721
6.602.3.3 doWriteArrayBounded	2721
6.602.3.4 doWriteByte	2721
6.602.3.5 flush	2721
6.602.3.6 lock	2721
6.602.3.7 notify	2722
6.602.3.8 notifyAll	2722
6.602.3.9 toString	2722
6.602.3.10tryLock	2723
6.602.3.11unlock	2723
6.602.3.12wait	2723
6.602.3.13wait	2723
6.602.3.14wait	2724
6.602.3.15write	2724
6.602.3.16write	2725
6.602.3.17write	2725
6.603decaf::io::OutputStreamWriter Class Reference	2726
6.603.1 Detailed Description	2726
6.603.2 Constructor & Destructor Documentation	2727
6.603.2.1 OutputStreamWriter	2727
6.603.2.2 ~OutputStreamWriter	2727
6.603.3 Member Function Documentation	2727
6.603.3.1 checkClosed	2727
6.603.3.2 close	2727
6.603.3.3 doWriteArrayBounded	2727
6.603.3.4 flush	2727
6.604activemq::commands::PartialCommand Class Reference	2727

6.604.1 Constructor & Destructor Documentation	2729
6.604.1.1 PartialCommand	2729
6.604.1.2 ~PartialCommand	2729
6.604.2 Member Function Documentation	2729
6.604.2.1 cloneDataStructure	2729
6.604.2.2 copyDataStructure	2729
6.604.2.3 equals	2729
6.604.2.4 getCommandId	2730
6.604.2.5 getData	2730
6.604.2.6 getData	2730
6.604.2.7 getDataStructureType	2730
6.604.2.8 setCommandId	2730
6.604.2.9 setData	2730
6.604.2.10 toString	2730
6.604.3 Field Documentation	2731
6.604.3.1 commandId	2731
6.604.3.2 data	2731
6.604.3.3 ID_PARTIALCOMMAND	2731
6.605activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller Class	
Reference	2731
6.605.1 Detailed Description	2732
6.605.2 Constructor & Destructor Documentation	2732
6.605.2.1 PartialCommandMarshaller	2732
6.605.2.2 ~PartialCommandMarshaller	2732
6.605.3 Member Function Documentation	2732
6.605.3.1 createObject	2732
6.605.3.2 getDataStructureType	2732
6.605.3.3 looseMarshal	2733
6.605.3.4 looseUnmarshal	2733
6.605.3.5 tightMarshal1	2734
6.605.3.6 tightMarshal2	2734
6.605.3.7 tightUnmarshal	2735
6.606activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller Class	
Reference	2735
6.606.1 Detailed Description	2736
6.606.2 Constructor & Destructor Documentation	2736
6.606.2.1 PartialCommandMarshaller	2736

6.606.2.2 ~PartialCommandMarshaller	2736
6.606.3 Member Function Documentation	2736
6.606.3.1 createObject	2736
6.606.3.2 getDataStructureType	2737
6.606.3.3 looseMarshal	2737
6.606.3.4 looseUnmarshal	2737
6.606.3.5 tightMarshal1	2738
6.606.3.6 tightMarshal2	2738
6.606.3.7 tightUnmarshal	2739
6.607activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller Class	
Reference	2739
6.607.1 Detailed Description	2740
6.607.2 Constructor & Destructor Documentation	2741
6.607.2.1 PartialCommandMarshaller	2741
6.607.2.2 ~PartialCommandMarshaller	2741
6.607.3 Member Function Documentation	2741
6.607.3.1 createObject	2741
6.607.3.2 getDataStructureType	2741
6.607.3.3 looseMarshal	2741
6.607.3.4 looseUnmarshal	2742
6.607.3.5 tightMarshal1	2742
6.607.3.6 tightMarshal2	2743
6.607.3.7 tightUnmarshal	2743
6.608activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller Class	
Reference	2744
6.608.1 Detailed Description	2745
6.608.2 Constructor & Destructor Documentation	2745
6.608.2.1 PartialCommandMarshaller	2745
6.608.2.2 ~PartialCommandMarshaller	2745
6.608.3 Member Function Documentation	2745
6.608.3.1 createObject	2745
6.608.3.2 getDataStructureType	2745
6.608.3.3 looseMarshal	2745
6.608.3.4 looseUnmarshal	2746
6.608.3.5 tightMarshal1	2746
6.608.3.6 tightMarshal2	2747
6.608.3.7 tightUnmarshal	2747

6.609	activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	Class	
	Reference		2748
6.609.1	Detailed Description		2749
6.609.2	Constructor & Destructor Documentation		2749
	6.609.2.1 PartialCommandMarshaller		2749
	6.609.2.2 ~PartialCommandMarshaller		2749
6.609.3	Member Function Documentation		2749
	6.609.3.1 createObject		2749
	6.609.3.2 getDataStructureType		2749
	6.609.3.3 looseMarshal		2750
	6.609.3.4 looseUnmarshal		2750
	6.609.3.5 tightMarshal1		2750
	6.609.3.6 tightMarshal2		2751
	6.609.3.7 tightUnmarshal		2751
6.610	activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	Class	
	Reference		2752
6.610.1	Detailed Description		2753
6.610.2	Constructor & Destructor Documentation		2753
	6.610.2.1 PartialCommandMarshaller		2753
	6.610.2.2 ~PartialCommandMarshaller		2753
6.610.3	Member Function Documentation		2753
	6.610.3.1 createObject		2753
	6.610.3.2 getDataStructureType		2753
	6.610.3.3 looseMarshal		2754
	6.610.3.4 looseUnmarshal		2754
	6.610.3.5 tightMarshal1		2755
	6.610.3.6 tightMarshal2		2755
	6.610.3.7 tightUnmarshal		2756
6.611	decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference		2756
	6.611.1 Detailed Description		2758
	6.611.2 Member Typedef Documentation		2758
	6.611.2.1 CounterType		2758
	6.611.2.2 PointerType		2758
	6.611.2.3 ReferenceType		2758
	6.611.3 Constructor & Destructor Documentation		2758
	6.611.3.1 Pointer		2758
	6.611.3.2 Pointer		2759

6.611.3.3 Pointer	2759
6.611.3.4 Pointer	2759
6.611.3.5 Pointer	2759
6.611.3.6 Pointer	2760
6.611.3.7 ~Pointer	2760
6.611.4 Member Function Documentation	2760
6.611.4.1 dynamicCast	2760
6.611.4.2 get	2760
6.611.4.3 operator!	2761
6.611.4.4 operator!=	2761
6.611.4.5 operator*	2761
6.611.4.6 operator*	2761
6.611.4.7 operator->	2761
6.611.4.8 operator->	2762
6.611.4.9 operator=	2762
6.611.4.10operator=	2762
6.611.4.11operator==	2762
6.611.4.12release	2762
6.611.4.13reset	2762
6.611.4.14staticCast	2763
6.611.4.15swap	2763
6.611.5 Friends And Related Function Documentation	2763
6.611.5.1 operator!=	2763
6.611.5.2 operator!=	2763
6.611.5.3 operator==	2763
6.611.5.4 operator==	2763
6.612decaf::lang::PointerComparator< T, R > Class Template Reference	2764
6.612.1 Detailed Description	2764
6.612.2 Member Function Documentation	2764
6.612.2.1 compare	2764
6.612.2.2 operator()	2764
6.613activemq::cmsutil::PooledSession Class Reference	2765
6.613.1 Detailed Description	2767
6.613.2 Constructor & Destructor Documentation	2767
6.613.2.1 PooledSession	2767
6.613.2.2 PooledSession	2767

6.613.2.3 ~PooledSession	2767
6.613.3 Member Function Documentation	2768
6.613.3.1 close	2768
6.613.3.2 commit	2768
6.613.3.3 createBrowser	2768
6.613.3.4 createBrowser	2768
6.613.3.5 createBytesMessage	2769
6.613.3.6 createBytesMessage	2769
6.613.3.7 createCachedConsumer	2769
6.613.3.8 createCachedProducer	2770
6.613.3.9 createConsumer	2770
6.613.3.10 createConsumer	2771
6.613.3.11 createConsumer	2771
6.613.3.12 createDurableConsumer	2772
6.613.3.13 createMapMessage	2772
6.613.3.14 createMessage	2772
6.613.3.15 createProducer	2773
6.613.3.16 createQueue	2773
6.613.3.17 createStreamMessage	2773
6.613.3.18 createTemporaryQueue	2774
6.613.3.19 createTemporaryTopic	2774
6.613.3.20 createTextMessage	2774
6.613.3.21 createTextMessage	2774
6.613.3.22 createTopic	2775
6.613.3.23 getAcknowledgeMode	2775
6.613.3.24 getSession	2775
6.613.3.25 getSession	2775
6.613.3.26 sTransacted	2776
6.613.3.27 operator=	2776
6.613.3.28 recover	2776
6.613.3.29 rollback	2776
6.613.3.30 unsubscribe	2777
6.614 decaf::util::concurrent::PooledThread Class Reference	2777
6.614.1 Constructor & Destructor Documentation	2778
6.614.1.1 PooledThread	2778
6.614.1.2 ~PooledThread	2778

6.614.2 Member Function Documentation	2778
6.614.2.1 getPooledThreadListener	2778
6.614.2.2 isBusy	2778
6.614.2.3 run	2778
6.614.2.4 setPooledThreadListener	2779
6.614.2.5 stop	2779
6.615decaf::util::concurrent::PooledThreadListener Class Reference	2779
6.615.1 Detailed Description	2780
6.615.2 Constructor & Destructor Documentation	2780
6.615.2.1 ~PooledThreadListener	2780
6.615.3 Member Function Documentation	2780
6.615.3.1 onTaskCompleted	2780
6.615.3.2 onTaskException	2780
6.615.3.3 onTaskStarted	2780
6.616decaf::net::PortUnreachableException Class Reference	2781
6.616.1 Constructor & Destructor Documentation	2781
6.616.1.1 PortUnreachableException	2781
6.616.1.2 PortUnreachableException	2782
6.616.1.3 PortUnreachableException	2782
6.616.1.4 PortUnreachableException	2782
6.616.1.5 PortUnreachableException	2782
6.616.1.6 PortUnreachableException	2782
6.616.1.7 ~PortUnreachableException	2783
6.616.2 Member Function Documentation	2783
6.616.2.1 clone	2783
6.617activemq::core::PrefetchPolicy Class Reference	2783
6.617.1 Detailed Description	2784
6.617.2 Constructor & Destructor Documentation	2785
6.617.2.1 PrefetchPolicy	2785
6.617.2.2 ~PrefetchPolicy	2785
6.617.3 Member Function Documentation	2785
6.617.3.1 clone	2785
6.617.3.2 configure	2785
6.617.3.3 getDurableTopicPrefetch	2785
6.617.3.4 getMaxPrefetchLimit	2786
6.617.3.5 getQueueBrowserPrefetch	2786

6.617.3.6	getQueuePrefetch	2786
6.617.3.7	getTopicPrefetch	2786
6.617.3.8	setDurableTopicPrefetch	2786
6.617.3.9	setQueueBrowserPrefetch	2787
6.617.3.10	setQueuePrefetch	2787
6.617.3.11	setTopicPrefetch	2787
6.618	activemq::util::PrimitiveList Class Reference	2787
6.618.1	Detailed Description	2790
6.618.2	Constructor & Destructor Documentation	2790
6.618.2.1	PrimitiveList	2790
6.618.2.2	~PrimitiveList	2790
6.618.2.3	PrimitiveList	2790
6.618.2.4	PrimitiveList	2790
6.618.3	Member Function Documentation	2791
6.618.3.1	getBool	2791
6.618.3.2	getByte	2791
6.618.3.3	getByteArray	2791
6.618.3.4	getChar	2792
6.618.3.5	getDouble	2792
6.618.3.6	getFloat	2793
6.618.3.7	getInt	2793
6.618.3.8	getLong	2793
6.618.3.9	getShort	2794
6.618.3.10	getString	2794
6.618.3.11	setBool	2794
6.618.3.12	setByte	2795
6.618.3.13	setByteArray	2795
6.618.3.14	setChar	2795
6.618.3.15	setDouble	2796
6.618.3.16	setFloat	2796
6.618.3.17	setInt	2796
6.618.3.18	setLong	2797
6.618.3.19	setShort	2797
6.618.3.20	setString	2797
6.618.3.21	toString	2798
6.619	activemq::util::PrimitiveMap Class Reference	2798

6.619.1 Detailed Description	2800
6.619.2 Constructor & Destructor Documentation	2800
6.619.2.1 PrimitiveMap	2800
6.619.2.2 ~PrimitiveMap	2801
6.619.2.3 PrimitiveMap	2801
6.619.2.4 PrimitiveMap	2801
6.619.3 Member Function Documentation	2801
6.619.3.1 getBool	2801
6.619.3.2 getByte	2801
6.619.3.3 getByteArray	2802
6.619.3.4 getChar	2802
6.619.3.5 getDouble	2803
6.619.3.6 getFloat	2803
6.619.3.7 getInt	2803
6.619.3.8 getLong	2804
6.619.3.9 getShort	2804
6.619.3.10 getString	2805
6.619.3.11 setBool	2805
6.619.3.12 setByte	2805
6.619.3.13 setByteArray	2805
6.619.3.14 setChar	2806
6.619.3.15 setDouble	2806
6.619.3.16 setFloat	2806
6.619.3.17 setInt	2806
6.619.3.18 setLong	2807
6.619.3.19 setShort	2807
6.619.3.20 setString	2807
6.619.3.21 toString	2807
6.620 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference	2808
6.620.1 Detailed Description	2809
6.620.2 Constructor & Destructor Documentation	2810
6.620.2.1 PrimitiveTypesMarshaller	2810
6.620.2.2 ~PrimitiveTypesMarshaller	2810
6.620.3 Member Function Documentation	2810
6.620.3.1 marshal	2810
6.620.3.2 marshal	2810

6.620.3.3 marshalList	2810
6.620.3.4 marshalMap	2811
6.620.3.5 marshalPrimitive	2811
6.620.3.6 marshalPrimitiveList	2811
6.620.3.7 marshalPrimitiveMap	2812
6.620.3.8 unmarshal	2812
6.620.3.9 unmarshal	2812
6.620.3.10 unmarshalList	2813
6.620.3.11 unmarshalMap	2813
6.620.3.12 unmarshalPrimitive	2813
6.620.3.13 unmarshalPrimitiveList	2814
6.620.3.14 unmarshalPrimitiveMap	2814
6.621 activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference	2814
6.621.1 Detailed Description	2815
6.621.2 Field Documentation	2815
6.621.2.1 boolValue	2815
6.621.2.2 byteArrayValue	2815
6.621.2.3 byteValue	2815
6.621.2.4 charValue	2815
6.621.2.5 doubleValue	2815
6.621.2.6 floatValue	2815
6.621.2.7 intValue	2815
6.621.2.8 listValue	2815
6.621.2.9 longValue	2815
6.621.2.10 mapValue	2815
6.621.2.11 shortValue	2815
6.621.2.12 stringValue	2815
6.622 activemq::util::PrimitiveValueConverter Class Reference	2816
6.622.1 Detailed Description	2816
6.622.2 Constructor & Destructor Documentation	2817
6.622.2.1 PrimitiveValueConverter	2817
6.622.2.2 ~PrimitiveValueConverter	2817
6.622.3 Member Function Documentation	2817
6.622.3.1 convert	2817
6.623 activemq::util::PrimitiveValueNode Class Reference	2817
6.623.1 Detailed Description	2821

6.623.2 Member Enumeration Documentation	2821
6.623.2.1 PrimitiveType	2821
6.623.3 Constructor & Destructor Documentation	2821
6.623.3.1 PrimitiveValueNode	2821
6.623.3.2 PrimitiveValueNode	2821
6.623.3.3 PrimitiveValueNode	2822
6.623.3.4 PrimitiveValueNode	2822
6.623.3.5 PrimitiveValueNode	2822
6.623.3.6 PrimitiveValueNode	2822
6.623.3.7 PrimitiveValueNode	2822
6.623.3.8 PrimitiveValueNode	2822
6.623.3.9 PrimitiveValueNode	2823
6.623.3.10 PrimitiveValueNode	2823
6.623.3.11 PrimitiveValueNode	2823
6.623.3.12 PrimitiveValueNode	2823
6.623.3.13 PrimitiveValueNode	2823
6.623.3.14 PrimitiveValueNode	2823
6.623.3.15 PrimitiveValueNode	2824
6.623.3.16 ~PrimitiveValueNode	2824
6.623.4 Member Function Documentation	2824
6.623.4.1 clear	2824
6.623.4.2 getBool	2824
6.623.4.3 getByte	2824
6.623.4.4 getByteArray	2824
6.623.4.5 getChar	2825
6.623.4.6 getDouble	2825
6.623.4.7 getFloat	2825
6.623.4.8 getInt	2825
6.623.4.9 getList	2826
6.623.4.10 getLong	2826
6.623.4.11 getMap	2826
6.623.4.12 getShort	2827
6.623.4.13 getString	2827
6.623.4.14 getType	2827
6.623.4.15 getValue	2827
6.623.4.16 operator=	2827

6.623.4.17	operator==	2828
6.623.4.18	setBool	2828
6.623.4.19	setByte	2828
6.623.4.20	setByteArray	2828
6.623.4.21	setChar	2828
6.623.4.22	setDouble	2829
6.623.4.23	setFloat	2829
6.623.4.24	setInt	2829
6.623.4.25	setList	2829
6.623.4.26	setLong	2829
6.623.4.27	setMap	2830
6.623.4.28	setShort	2830
6.623.4.29	setString	2830
6.623.4.30	setValue	2830
6.623.4.31	toString	2830
6.624	decaf::security::Principal Class Reference	2831
6.624.1	Detailed Description	2831
6.624.2	Constructor & Destructor Documentation	2831
6.624.2.1	~Principal	2831
6.624.3	Member Function Documentation	2831
6.624.3.1	equals	2831
6.624.3.2	getName	2831
6.625	decaf::util::PriorityQueue< E > Class Template Reference	2832
6.625.1	Detailed Description	2833
6.625.2	Constructor & Destructor Documentation	2834
6.625.2.1	PriorityQueue	2834
6.625.2.2	PriorityQueue	2834
6.625.2.3	PriorityQueue	2834
6.625.2.4	PriorityQueue	2835
6.625.2.5	PriorityQueue	2835
6.625.2.6	~PriorityQueue	2835
6.625.3	Member Function Documentation	2835
6.625.3.1	add	2835
6.625.3.2	clear	2836
6.625.3.3	comparator	2836
6.625.3.4	iterator	2836

6.625.3.5 iterator	2836
6.625.3.6 offer	2836
6.625.3.7 operator=	2837
6.625.3.8 operator=	2837
6.625.3.9 peek	2837
6.625.3.10 poll	2837
6.625.3.11 remove	2838
6.625.3.12 remove	2838
6.625.3.13 size	2839
6.625.4 Friends And Related Function Documentation	2839
6.625.4.1 PriorityQueueIterator	2839
6.626 activemq::commands::ProducerAck Class Reference	2839
6.626.1 Constructor & Destructor Documentation	2840
6.626.1.1 ProducerAck	2840
6.626.1.2 ~ProducerAck	2840
6.626.2 Member Function Documentation	2840
6.626.2.1 cloneDataStructure	2840
6.626.2.2 copyDataStructure	2841
6.626.2.3 equals	2841
6.626.2.4 getDataStructureType	2841
6.626.2.5 getProducerId	2841
6.626.2.6 getProducerId	2841
6.626.2.7 getSize	2841
6.626.2.8 isProducerAck	2841
6.626.2.9 setProducerId	2842
6.626.2.10 setSize	2842
6.626.2.11 toString	2842
6.626.2.12 visit	2842
6.626.3 Field Documentation	2842
6.626.3.1 ID_PRODUCERACK	2842
6.626.3.2 producerId	2842
6.626.3.3 size	2842
6.627 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class Reference	2842
6.627.1 Detailed Description	2843
6.627.2 Constructor & Destructor Documentation	2844
6.627.2.1 ProducerAckMarshaller	2844

6.627.2.2 ~ProducerAckMarshaller	2844
6.627.3 Member Function Documentation	2844
6.627.3.1 createObject	2844
6.627.3.2 getDataStructureType	2844
6.627.3.3 looseMarshal	2844
6.627.3.4 looseUnmarshal	2845
6.627.3.5 tightMarshal1	2845
6.627.3.6 tightMarshal2	2845
6.627.3.7 tightUnmarshal	2846
6.628activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller Class Reference	2846
6.628.1 Detailed Description	2847
6.628.2 Constructor & Destructor Documentation	2848
6.628.2.1 ProducerAckMarshaller	2848
6.628.2.2 ~ProducerAckMarshaller	2848
6.628.3 Member Function Documentation	2848
6.628.3.1 createObject	2848
6.628.3.2 getDataStructureType	2848
6.628.3.3 looseMarshal	2848
6.628.3.4 looseUnmarshal	2849
6.628.3.5 tightMarshal1	2849
6.628.3.6 tightMarshal2	2849
6.628.3.7 tightUnmarshal	2850
6.629activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller Class Reference	2850
6.629.1 Detailed Description	2851
6.629.2 Constructor & Destructor Documentation	2852
6.629.2.1 ProducerAckMarshaller	2852
6.629.2.2 ~ProducerAckMarshaller	2852
6.629.3 Member Function Documentation	2852
6.629.3.1 createObject	2852
6.629.3.2 getDataStructureType	2852
6.629.3.3 looseMarshal	2852
6.629.3.4 looseUnmarshal	2853
6.629.3.5 tightMarshal1	2853
6.629.3.6 tightMarshal2	2853
6.629.3.7 tightUnmarshal	2854
6.630activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller Class Reference	2854

6.630.1 Detailed Description	2855
6.630.2 Constructor & Destructor Documentation	2856
6.630.2.1 ProducerAckMarshaller	2856
6.630.2.2 ~ProducerAckMarshaller	2856
6.630.3 Member Function Documentation	2856
6.630.3.1 createObject	2856
6.630.3.2 getDataStructureType	2856
6.630.3.3 looseMarshal	2856
6.630.3.4 looseUnmarshal	2857
6.630.3.5 tightMarshal1	2857
6.630.3.6 tightMarshal2	2857
6.630.3.7 tightUnmarshal	2858
6.631activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller Class Reference	2858
6.631.1 Detailed Description	2859
6.631.2 Constructor & Destructor Documentation	2860
6.631.2.1 ProducerAckMarshaller	2860
6.631.2.2 ~ProducerAckMarshaller	2860
6.631.3 Member Function Documentation	2860
6.631.3.1 createObject	2860
6.631.3.2 getDataStructureType	2860
6.631.3.3 looseMarshal	2860
6.631.3.4 looseUnmarshal	2861
6.631.3.5 tightMarshal1	2861
6.631.3.6 tightMarshal2	2861
6.631.3.7 tightUnmarshal	2862
6.632activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller Class Reference	2862
6.632.1 Detailed Description	2863
6.632.2 Constructor & Destructor Documentation	2864
6.632.2.1 ProducerAckMarshaller	2864
6.632.2.2 ~ProducerAckMarshaller	2864
6.632.3 Member Function Documentation	2864
6.632.3.1 createObject	2864
6.632.3.2 getDataStructureType	2864
6.632.3.3 looseMarshal	2864
6.632.3.4 looseUnmarshal	2865
6.632.3.5 tightMarshal1	2865

6.632.3.6 tightMarshal2	2865
6.632.3.7 tightUnmarshal	2866
6.633activemq::cmsutil::ProducerCallback Class Reference	2866
6.633.1 Detailed Description	2867
6.633.2 Constructor & Destructor Documentation	2867
6.633.2.1 ~ProducerCallback	2867
6.633.3 Member Function Documentation	2867
6.633.3.1 doInCms	2867
6.634activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference	2867
6.634.1 Constructor & Destructor Documentation	2868
6.634.1.1 ProducerExecutor	2868
6.634.1.2 ProducerExecutor	2868
6.634.1.3 ~ProducerExecutor	2868
6.634.2 Member Function Documentation	2868
6.634.2.1 doInCms	2868
6.634.2.2 getDestination	2869
6.634.2.3 operator=	2869
6.634.3 Field Documentation	2869
6.634.3.1 action	2869
6.634.3.2 destination	2869
6.634.3.3 parent	2869
6.635activemq::commands::ProducerId Class Reference	2869
6.635.1 Member Typedef Documentation	2871
6.635.1.1 COMPARATOR	2871
6.635.2 Constructor & Destructor Documentation	2871
6.635.2.1 ProducerId	2871
6.635.2.2 ProducerId	2871
6.635.2.3 ProducerId	2871
6.635.2.4 ProducerId	2871
6.635.2.5 ~ProducerId	2871
6.635.3 Member Function Documentation	2871
6.635.3.1 cloneDataStructure	2871
6.635.3.2 compareTo	2871
6.635.3.3 copyDataStructure	2871
6.635.3.4 equals	2871
6.635.3.5 equals	2872

6.635.3.6	getConnectionId	2872
6.635.3.7	getConnectionId	2872
6.635.3.8	getDataStructureType	2872
6.635.3.9	getParentId	2873
6.635.3.10	getSessionId	2873
6.635.3.11	getValue	2873
6.635.3.12	operator<	2873
6.635.3.13	operator=	2873
6.635.3.14	operator==	2873
6.635.3.15	setConnectionId	2873
6.635.3.16	setProducerSessionKey	2873
6.635.3.17	setSessionId	2873
6.635.3.18	setValue	2873
6.635.3.19	toString	2873
6.635.4	Field Documentation	2873
6.635.4.1	connectionId	2873
6.635.4.2	ID_PRODUCERID	2873
6.635.4.3	sessionId	2874
6.635.4.4	value	2874
6.636	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller Class Reference	2874
6.636.1	Detailed Description	2875
6.636.2	Constructor & Destructor Documentation	2875
6.636.2.1	ProducerIdMarshaller	2875
6.636.2.2	~ProducerIdMarshaller	2875
6.636.3	Member Function Documentation	2875
6.636.3.1	createObject	2875
6.636.3.2	getDataStructureType	2875
6.636.3.3	looseMarshal	2876
6.636.3.4	looseUnmarshal	2876
6.636.3.5	tightMarshal1	2876
6.636.3.6	tightMarshal2	2877
6.636.3.7	tightUnmarshal	2877
6.637	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller Class Reference	2878
6.637.1	Detailed Description	2879
6.637.2	Constructor & Destructor Documentation	2879
6.637.2.1	ProducerIdMarshaller	2879

6.637.2.2 ~ProducerIdMarshaller	2879
6.637.3 Member Function Documentation	2879
6.637.3.1 createObject	2879
6.637.3.2 getDataStructureType	2879
6.637.3.3 looseMarshal	2879
6.637.3.4 looseUnmarshal	2880
6.637.3.5 tightMarshal1	2880
6.637.3.6 tightMarshal2	2881
6.637.3.7 tightUnmarshal	2881
6.638activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller Class Reference	2881
6.638.1 Detailed Description	2882
6.638.2 Constructor & Destructor Documentation	2883
6.638.2.1 ProducerIdMarshaller	2883
6.638.2.2 ~ProducerIdMarshaller	2883
6.638.3 Member Function Documentation	2883
6.638.3.1 createObject	2883
6.638.3.2 getDataStructureType	2883
6.638.3.3 looseMarshal	2883
6.638.3.4 looseUnmarshal	2884
6.638.3.5 tightMarshal1	2884
6.638.3.6 tightMarshal2	2884
6.638.3.7 tightUnmarshal	2885
6.639activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class Reference	2885
6.639.1 Detailed Description	2886
6.639.2 Constructor & Destructor Documentation	2887
6.639.2.1 ProducerIdMarshaller	2887
6.639.2.2 ~ProducerIdMarshaller	2887
6.639.3 Member Function Documentation	2887
6.639.3.1 createObject	2887
6.639.3.2 getDataStructureType	2887
6.639.3.3 looseMarshal	2887
6.639.3.4 looseUnmarshal	2888
6.639.3.5 tightMarshal1	2888
6.639.3.6 tightMarshal2	2888
6.639.3.7 tightUnmarshal	2889
6.640activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller Class Reference	2889

6.640.1 Detailed Description	2890
6.640.2 Constructor & Destructor Documentation	2891
6.640.2.1 ProducerIdMarshaller	2891
6.640.2.2 ~ProducerIdMarshaller	2891
6.640.3 Member Function Documentation	2891
6.640.3.1 createObject	2891
6.640.3.2 getDataStructureType	2891
6.640.3.3 looseMarshal	2891
6.640.3.4 looseUnmarshal	2892
6.640.3.5 tightMarshal1	2892
6.640.3.6 tightMarshal2	2892
6.640.3.7 tightUnmarshal	2893
6.641activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller Class Reference	2893
6.641.1 Detailed Description	2894
6.641.2 Constructor & Destructor Documentation	2895
6.641.2.1 ProducerIdMarshaller	2895
6.641.2.2 ~ProducerIdMarshaller	2895
6.641.3 Member Function Documentation	2895
6.641.3.1 createObject	2895
6.641.3.2 getDataStructureType	2895
6.641.3.3 looseMarshal	2895
6.641.3.4 looseUnmarshal	2896
6.641.3.5 tightMarshal1	2896
6.641.3.6 tightMarshal2	2896
6.641.3.7 tightUnmarshal	2897
6.642activemq::commands::ProducerInfo Class Reference	2897
6.642.1 Constructor & Destructor Documentation	2899
6.642.1.1 ProducerInfo	2899
6.642.1.2 ~ProducerInfo	2899
6.642.2 Member Function Documentation	2899
6.642.2.1 cloneDataStructure	2899
6.642.2.2 copyDataStructure	2899
6.642.2.3 createRemoveCommand	2899
6.642.2.4 equals	2899
6.642.2.5 getBrokerPath	2900
6.642.2.6 getBrokerPath	2900

6.642.2.7	getDataStructureType	2900
6.642.2.8	getDestination	2900
6.642.2.9	getDestination	2900
6.642.2.10	getProducerId	2900
6.642.2.11	getProducerId	2900
6.642.2.12	getWindowSize	2900
6.642.2.13	sDispatchAsync	2900
6.642.2.14	sProducerInfo	2900
6.642.2.15	setBrokerPath	2901
6.642.2.16	setDestination	2901
6.642.2.17	setDispatchAsync	2901
6.642.2.18	setProducerId	2901
6.642.2.19	setWindowSize	2901
6.642.2.20	toString	2901
6.642.2.21	visit	2901
6.642.3	Field Documentation	2902
6.642.3.1	brokerPath	2902
6.642.3.2	destination	2902
6.642.3.3	dispatchAsync	2902
6.642.3.4	ID_PRODUCERINFO	2902
6.642.3.5	producerId	2902
6.642.3.6	windowSize	2902
6.643	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller Class Reference	2902
6.643.1	Detailed Description	2903
6.643.2	Constructor & Destructor Documentation	2903
6.643.2.1	ProducerInfoMarshaller	2903
6.643.2.2	~ProducerInfoMarshaller	2903
6.643.3	Member Function Documentation	2903
6.643.3.1	createObject	2903
6.643.3.2	getDataStructureType	2903
6.643.3.3	looseMarshal	2904
6.643.3.4	looseUnmarshal	2904
6.643.3.5	tightMarshal1	2904
6.643.3.6	tightMarshal2	2905
6.643.3.7	tightUnmarshal	2905

6.644activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller Class Reference	2906
6.644.1 Detailed Description	2907
6.644.2 Constructor & Destructor Documentation	2907
6.644.2.1 ProducerInfoMarshaller	2907
6.644.2.2 ~ProducerInfoMarshaller	2907
6.644.3 Member Function Documentation	2907
6.644.3.1 createObject	2907
6.644.3.2 getDataStructureType	2907
6.644.3.3 looseMarshal	2908
6.644.3.4 looseUnmarshal	2908
6.644.3.5 tightMarshal1	2908
6.644.3.6 tightMarshal2	2909
6.644.3.7 tightUnmarshal	2909
6.645activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller Class Reference	2910
6.645.1 Detailed Description	2911
6.645.2 Constructor & Destructor Documentation	2911
6.645.2.1 ProducerInfoMarshaller	2911
6.645.2.2 ~ProducerInfoMarshaller	2911
6.645.3 Member Function Documentation	2911
6.645.3.1 createObject	2911
6.645.3.2 getDataStructureType	2911
6.645.3.3 looseMarshal	2912
6.645.3.4 looseUnmarshal	2912
6.645.3.5 tightMarshal1	2912
6.645.3.6 tightMarshal2	2913
6.645.3.7 tightUnmarshal	2913
6.646activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller Class Reference	2914
6.646.1 Detailed Description	2915
6.646.2 Constructor & Destructor Documentation	2915
6.646.2.1 ProducerInfoMarshaller	2915
6.646.2.2 ~ProducerInfoMarshaller	2915
6.646.3 Member Function Documentation	2915
6.646.3.1 createObject	2915
6.646.3.2 getDataStructureType	2915

6.646.3.3 looseMarshal	2916
6.646.3.4 looseUnmarshal	2916
6.646.3.5 tightMarshal1	2916
6.646.3.6 tightMarshal2	2917
6.646.3.7 tightUnmarshal	2917
6.647activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller Class Reference	2918
6.647.1 Detailed Description	2919
6.647.2 Constructor & Destructor Documentation	2919
6.647.2.1 ProducerInfoMarshaller	2919
6.647.2.2 ~ProducerInfoMarshaller	2919
6.647.3 Member Function Documentation	2919
6.647.3.1 createObject	2919
6.647.3.2 getDataStructureType	2919
6.647.3.3 looseMarshal	2920
6.647.3.4 looseUnmarshal	2920
6.647.3.5 tightMarshal1	2920
6.647.3.6 tightMarshal2	2921
6.647.3.7 tightUnmarshal	2921
6.648activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller Class Reference	2922
6.648.1 Detailed Description	2923
6.648.2 Constructor & Destructor Documentation	2923
6.648.2.1 ProducerInfoMarshaller	2923
6.648.2.2 ~ProducerInfoMarshaller	2923
6.648.3 Member Function Documentation	2923
6.648.3.1 createObject	2923
6.648.3.2 getDataStructureType	2923
6.648.3.3 looseMarshal	2924
6.648.3.4 looseUnmarshal	2924
6.648.3.5 tightMarshal1	2924
6.648.3.6 tightMarshal2	2925
6.648.3.7 tightUnmarshal	2925
6.649activemq::state::ProducerState Class Reference	2926
6.649.1 Constructor & Destructor Documentation	2926
6.649.1.1 ProducerState	2926
6.649.1.2 ~ProducerState	2926

6.649.2 Member Function Documentation	2926
6.649.2.1 getInfo	2926
6.649.2.2 getTransactionState	2926
6.649.2.3 setTransactionState	2926
6.649.2.4 toString	2926
6.650decaf::util::Properties Class Reference	2927
6.650.1 Detailed Description	2928
6.650.2 Constructor & Destructor Documentation	2929
6.650.2.1 Properties	2929
6.650.2.2 Properties	2929
6.650.2.3 ~Properties	2929
6.650.3 Member Function Documentation	2929
6.650.3.1 clear	2929
6.650.3.2 clone	2929
6.650.3.3 copy	2929
6.650.3.4 equals	2929
6.650.3.5 getProperty	2930
6.650.3.6 getProperty	2930
6.650.3.7 hasProperty	2930
6.650.3.8 isEmpty	2930
6.650.3.9 load	2931
6.650.3.10load	2931
6.650.3.11operator=	2933
6.650.3.12propertyName	2933
6.650.3.13remove	2933
6.650.3.14setProperty	2933
6.650.3.15size	2934
6.650.3.16store	2934
6.650.3.17store	2934
6.650.3.18oArray	2935
6.650.3.19oString	2935
6.650.4 Field Documentation	2936
6.650.4.1 defaults	2936
6.651decaf::util::logging::PropertiesChangeListener Class Reference	2936
6.651.1 Detailed Description	2936
6.651.2 Constructor & Destructor Documentation	2937

6.651.2.1 ~PropertiesChangeListener	2937
6.651.3 Member Function Documentation	2937
6.651.3.1 onPropertiesReset	2937
6.651.3.2 onPropertyChanged	2937
6.652decaf::net::ProtocolException Class Reference	2937
6.652.1 Constructor & Destructor Documentation	2938
6.652.1.1 ProtocolException	2938
6.652.1.2 ProtocolException	2938
6.652.1.3 ProtocolException	2938
6.652.1.4 ProtocolException	2938
6.652.1.5 ProtocolException	2939
6.652.1.6 ProtocolException	2939
6.652.1.7 ~ProtocolException	2939
6.652.2 Member Function Documentation	2939
6.652.2.1 clone	2939
6.653decaf::security::PublicKey Class Reference	2940
6.653.1 Detailed Description	2940
6.653.2 Constructor & Destructor Documentation	2940
6.653.2.1 ~PublicKey	2940
6.654decaf::io::PushbackInputStream Class Reference	2940
6.654.1 Detailed Description	2942
6.654.2 Constructor & Destructor Documentation	2942
6.654.2.1 PushbackInputStream	2942
6.654.2.2 PushbackInputStream	2942
6.654.2.3 ~PushbackInputStream	2943
6.654.3 Member Function Documentation	2943
6.654.3.1 available	2943
6.654.3.2 doReadArrayBounded	2943
6.654.3.3 doReadByte	2943
6.654.3.4 mark	2944
6.654.3.5 markSupported	2944
6.654.3.6 reset	2944
6.654.3.7 skip	2945
6.654.3.8 unread	2945
6.654.3.9 unread	2946
6.654.3.10unread	2946

6.655	cms::Queue Class Reference	2947
6.655.1	Detailed Description	2947
6.655.2	Constructor & Destructor Documentation	2947
6.655.2.1	~Queue	2947
6.655.3	Member Function Documentation	2947
6.655.3.1	getQueueName	2947
6.656	decaf::util::Queue< E > Class Template Reference	2948
6.656.1	Detailed Description	2948
6.656.2	Constructor & Destructor Documentation	2949
6.656.2.1	~Queue	2949
6.656.3	Member Function Documentation	2949
6.656.3.1	element	2949
6.656.3.2	offer	2949
6.656.3.3	peek	2950
6.656.3.4	poll	2950
6.656.3.5	remove	2950
6.657	cms::QueueBrowser Class Reference	2951
6.657.1	Detailed Description	2951
6.657.2	Constructor & Destructor Documentation	2952
6.657.2.1	~QueueBrowser	2952
6.657.3	Member Function Documentation	2952
6.657.3.1	getEnumeration	2952
6.657.3.2	getMessageSelector	2952
6.657.3.3	getQueue	2952
6.658	decaf::util::Random Class Reference	2953
6.658.1	Detailed Description	2954
6.658.2	Constructor & Destructor Documentation	2954
6.658.2.1	Random	2954
6.658.2.2	Random	2954
6.658.3	Member Function Documentation	2954
6.658.3.1	next	2954
6.658.3.2	nextBoolean	2955
6.658.3.3	nextBytes	2955
6.658.3.4	nextBytes	2955
6.658.3.5	nextDouble	2956
6.658.3.6	nextFloat	2956

6.658.3.7	nextGaussian	2956
6.658.3.8	nextInt	2956
6.658.3.9	nextInt	2957
6.658.3.10	nextLong	2957
6.658.3.11	setSeed	2957
6.659	decaf::lang::Readable Class Reference	2958
6.659.1	Detailed Description	2958
6.659.2	Constructor & Destructor Documentation	2958
6.659.2.1	~Readable	2958
6.659.3	Member Function Documentation	2958
6.659.3.1	read	2958
6.660	activemq::transport::inactivity::ReadChecker Class Reference	2959
6.660.1	Detailed Description	2959
6.660.2	Constructor & Destructor Documentation	2960
6.660.2.1	ReadChecker	2960
6.660.2.2	~ReadChecker	2960
6.660.3	Member Function Documentation	2960
6.660.3.1	run	2960
6.661	decaf::io::Reader Class Reference	2960
6.661.1	Constructor & Destructor Documentation	2962
6.661.1.1	Reader	2962
6.661.1.2	~Reader	2962
6.661.2	Member Function Documentation	2962
6.661.2.1	doReadArray	2962
6.661.2.2	doReadArrayBounded	2962
6.661.2.3	doReadChar	2962
6.661.2.4	doReadCharBuffer	2962
6.661.2.5	doReadVector	2962
6.661.2.6	mark	2962
6.661.2.7	markSupported	2963
6.661.2.8	read	2963
6.661.2.9	read	2964
6.661.2.10	read	2964
6.661.2.11	read	2964
6.661.2.12	read	2965
6.661.2.13	ready	2965

6.661.2.14	reset	2965
6.661.2.15	skip	2966
6.662	decaf::nio::ReadOnlyBufferException Class Reference	2966
6.662.1	Constructor & Destructor Documentation	2967
6.662.1.1	ReadOnlyBufferException	2967
6.662.1.2	ReadOnlyBufferException	2967
6.662.1.3	ReadOnlyBufferException	2967
6.662.1.4	ReadOnlyBufferException	2967
6.662.1.5	ReadOnlyBufferException	2968
6.662.1.6	ReadOnlyBufferException	2968
6.662.1.7	~ReadOnlyBufferException	2968
6.662.2	Member Function Documentation	2968
6.662.2.1	clone	2968
6.663	decaf::util::concurrent::locks::ReadWriteLock Class Reference	2968
6.663.1	Detailed Description	2969
6.663.2	Constructor & Destructor Documentation	2970
6.663.2.1	~ReadWriteLock	2970
6.663.3	Member Function Documentation	2970
6.663.3.1	readLock	2970
6.663.3.2	writeLock	2970
6.664	activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference	2970
6.664.1	Constructor & Destructor Documentation	2971
6.664.1.1	ReceiveExecutor	2971
6.664.1.2	ReceiveExecutor	2971
6.664.1.3	~ReceiveExecutor	2971
6.664.2	Member Function Documentation	2971
6.664.2.1	doInCms	2971
6.664.2.2	getDestination	2972
6.664.2.3	getMessage	2972
6.664.2.4	operator=	2972
6.664.3	Field Documentation	2972
6.664.3.1	destination	2972
6.664.3.2	message	2972
6.664.3.3	noLocal	2972
6.664.3.4	parent	2972
6.664.3.5	selector	2972

6.665	activemq::core::RedeliveryPolicy Class Reference	2972
6.665.1	Detailed Description	2974
6.665.2	Constructor & Destructor Documentation	2974
6.665.2.1	RedeliveryPolicy	2974
6.665.2.2	~RedeliveryPolicy	2974
6.665.3	Member Function Documentation	2974
6.665.3.1	clone	2974
6.665.3.2	configure	2974
6.665.3.3	getBackOffMultiplier	2975
6.665.3.4	getCollisionAvoidancePercent	2975
6.665.3.5	getInitialRedeliveryDelay	2975
6.665.3.6	getMaximumRedeliveries	2975
6.665.3.7	getRedeliveryDelay	2975
6.665.3.8	isUseCollisionAvoidance	2976
6.665.3.9	isUseExponentialBackOff	2976
6.665.3.10	setBackOffMultiplier	2976
6.665.3.11	setCollisionAvoidancePercent	2976
6.665.3.12	setInitialRedeliveryDelay	2976
6.665.3.13	setMaximumRedeliveries	2977
6.665.3.14	setUseCollisionAvoidance	2977
6.665.3.15	setUseExponentialBackOff	2977
6.665.4	Field Documentation	2977
6.665.4.1	NO_MAXIMUM_REDELIVERIES	2977
6.666	decaf::util::concurrent::locks::ReentrantLock Class Reference	2977
6.666.1	Detailed Description	2978
6.666.2	Constructor & Destructor Documentation	2979
6.666.2.1	ReentrantLock	2979
6.666.2.2	~ReentrantLock	2979
6.666.3	Member Function Documentation	2979
6.666.3.1	getHoldCount	2979
6.666.3.2	isFair	2980
6.666.3.3	isHeldByCurrentThread	2980
6.666.3.4	isLocked	2980
6.666.3.5	lock	2980
6.666.3.6	lockInterruptibly	2981
6.666.3.7	newCondition	2981

6.666.3.8 toString	2982
6.666.3.9 tryLock	2982
6.666.3.10 tryLock	2983
6.666.3.11 unlock	2984
6.667 decaf::util::concurrent::RejectedExecutionException Class Reference	2984
6.667.1 Constructor & Destructor Documentation	2985
6.667.1.1 RejectedExecutionException	2985
6.667.1.2 RejectedExecutionException	2985
6.667.1.3 RejectedExecutionException	2985
6.667.1.4 RejectedExecutionException	2985
6.667.1.5 RejectedExecutionException	2986
6.667.1.6 RejectedExecutionException	2986
6.667.1.7 ~RejectedExecutionException	2986
6.667.2 Member Function Documentation	2986
6.667.2.1 clone	2986
6.668 decaf::util::concurrent::RejectedExecutionHandler Class Reference	2987
6.668.1 Detailed Description	2987
6.668.2 Constructor & Destructor Documentation	2987
6.668.2.1 ~RejectedExecutionHandler	2987
6.668.3 Member Function Documentation	2987
6.668.3.1 rejectedExecution	2987
6.669 activemq::commands::RemoveInfo Class Reference	2988
6.669.1 Constructor & Destructor Documentation	2989
6.669.1.1 RemoveInfo	2989
6.669.1.2 ~RemoveInfo	2989
6.669.2 Member Function Documentation	2989
6.669.2.1 cloneDataStructure	2989
6.669.2.2 copyDataStructure	2989
6.669.2.3 equals	2989
6.669.2.4 getDataStructureType	2990
6.669.2.5 getLastDeliveredSequenceId	2990
6.669.2.6 getObjectId	2990
6.669.2.7 getObjectId	2990
6.669.2.8 isRemoveInfo	2990
6.669.2.9 setLastDeliveredSequenceId	2990
6.669.2.10 setObjectId	2990

6.669.2.1	toString	2990
6.669.2.12	visit	2991
6.669.3	Field Documentation	2991
6.669.3.1	ID_REMOVEINFO	2991
6.669.3.2	lastDeliveredSequenceId	2991
6.669.3.3	objectId	2991
6.670	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class Reference	2991
6.670.1	Detailed Description	2992
6.670.2	Constructor & Destructor Documentation	2992
6.670.2.1	RemoveInfoMarshaller	2992
6.670.2.2	~RemoveInfoMarshaller	2992
6.670.3	Member Function Documentation	2992
6.670.3.1	createObject	2992
6.670.3.2	getDataStructureType	2993
6.670.3.3	looseMarshal	2993
6.670.3.4	looseUnmarshal	2993
6.670.3.5	tightMarshal1	2994
6.670.3.6	tightMarshal2	2994
6.670.3.7	tightUnmarshal	2994
6.671	activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller Class Reference	2995
6.671.1	Detailed Description	2996
6.671.2	Constructor & Destructor Documentation	2996
6.671.2.1	RemoveInfoMarshaller	2996
6.671.2.2	~RemoveInfoMarshaller	2996
6.671.3	Member Function Documentation	2996
6.671.3.1	createObject	2996
6.671.3.2	getDataStructureType	2996
6.671.3.3	looseMarshal	2997
6.671.3.4	looseUnmarshal	2997
6.671.3.5	tightMarshal1	2997
6.671.3.6	tightMarshal2	2998
6.671.3.7	tightUnmarshal	2998
6.672	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller Class Reference	2999
6.672.1	Detailed Description	3000
6.672.2	Constructor & Destructor Documentation	3000
6.672.2.1	RemoveInfoMarshaller	3000

6.672.2.2 ~RemoveInfoMarshaller	3000
6.672.3 Member Function Documentation	3000
6.672.3.1 createObject	3000
6.672.3.2 getDataStructureType	3000
6.672.3.3 looseMarshal	3001
6.672.3.4 looseUnmarshal	3001
6.672.3.5 tightMarshal1	3001
6.672.3.6 tightMarshal2	3002
6.672.3.7 tightUnmarshal	3002
6.673activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller Class Reference	3003
6.673.1 Detailed Description	3004
6.673.2 Constructor & Destructor Documentation	3004
6.673.2.1 RemoveInfoMarshaller	3004
6.673.2.2 ~RemoveInfoMarshaller	3004
6.673.3 Member Function Documentation	3004
6.673.3.1 createObject	3004
6.673.3.2 getDataStructureType	3004
6.673.3.3 looseMarshal	3005
6.673.3.4 looseUnmarshal	3005
6.673.3.5 tightMarshal1	3005
6.673.3.6 tightMarshal2	3006
6.673.3.7 tightUnmarshal	3006
6.674activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller Class Reference	3007
6.674.1 Detailed Description	3008
6.674.2 Constructor & Destructor Documentation	3008
6.674.2.1 RemoveInfoMarshaller	3008
6.674.2.2 ~RemoveInfoMarshaller	3008
6.674.3 Member Function Documentation	3008
6.674.3.1 createObject	3008
6.674.3.2 getDataStructureType	3008
6.674.3.3 looseMarshal	3009
6.674.3.4 looseUnmarshal	3009
6.674.3.5 tightMarshal1	3009
6.674.3.6 tightMarshal2	3010
6.674.3.7 tightUnmarshal	3010
6.675activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller Class Reference	3011

6.675.1 Detailed Description	3012
6.675.2 Constructor & Destructor Documentation	3012
6.675.2.1 RemoveInfoMarshaller	3012
6.675.2.2 ~RemoveInfoMarshaller	3012
6.675.3 Member Function Documentation	3012
6.675.3.1 createObject	3012
6.675.3.2 getDataStructureType	3012
6.675.3.3 looseMarshal	3013
6.675.3.4 looseUnmarshal	3013
6.675.3.5 tightMarshal1	3013
6.675.3.6 tightMarshal2	3014
6.675.3.7 tightUnmarshal	3014
6.676activemq::commands::RemoveSubscriptionInfo Class Reference	3015
6.676.1 Constructor & Destructor Documentation	3016
6.676.1.1 RemoveSubscriptionInfo	3016
6.676.1.2 ~RemoveSubscriptionInfo	3016
6.676.2 Member Function Documentation	3016
6.676.2.1 cloneDataStructure	3016
6.676.2.2 copyDataStructure	3016
6.676.2.3 equals	3017
6.676.2.4 getClientId	3017
6.676.2.5 getClientId	3017
6.676.2.6 getConnectionId	3017
6.676.2.7 getConnectionId	3017
6.676.2.8 getDataStructureType	3017
6.676.2.9 getSubscriptionName	3018
6.676.2.10getSubscriptionName	3018
6.676.2.11isRemoveSubscriptionInfo	3018
6.676.2.12setClientId	3018
6.676.2.13setConnectionId	3018
6.676.2.14setSubscriptionName	3018
6.676.2.15toString	3018
6.676.2.16visit	3018
6.676.3 Field Documentation	3019
6.676.3.1 clientId	3019
6.676.3.2 connectionId	3019

6.676.3.3 ID_REMOVESUBSCRIPTIONINFO	3019
6.676.3.4 subscriptionName	3019
6.677activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	
Class Reference	3019
6.677.1 Detailed Description	3020
6.677.2 Constructor & Destructor Documentation	3020
6.677.2.1 RemoveSubscriptionInfoMarshaller	3020
6.677.2.2 ~RemoveSubscriptionInfoMarshaller	3020
6.677.3 Member Function Documentation	3020
6.677.3.1 createObject	3020
6.677.3.2 getDataStructureType	3021
6.677.3.3 looseMarshal	3021
6.677.3.4 looseUnmarshal	3021
6.677.3.5 tightMarshal1	3022
6.677.3.6 tightMarshal2	3022
6.677.3.7 tightUnmarshal	3022
6.678activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller	
Class Reference	3023
6.678.1 Detailed Description	3024
6.678.2 Constructor & Destructor Documentation	3024
6.678.2.1 RemoveSubscriptionInfoMarshaller	3024
6.678.2.2 ~RemoveSubscriptionInfoMarshaller	3024
6.678.3 Member Function Documentation	3024
6.678.3.1 createObject	3024
6.678.3.2 getDataStructureType	3024
6.678.3.3 looseMarshal	3025
6.678.3.4 looseUnmarshal	3025
6.678.3.5 tightMarshal1	3025
6.678.3.6 tightMarshal2	3026
6.678.3.7 tightUnmarshal	3026
6.679activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller	
Class Reference	3027
6.679.1 Detailed Description	3028
6.679.2 Constructor & Destructor Documentation	3028
6.679.2.1 RemoveSubscriptionInfoMarshaller	3028
6.679.2.2 ~RemoveSubscriptionInfoMarshaller	3028
6.679.3 Member Function Documentation	3028

6.679.3.1	createObject	3028
6.679.3.2	getDataStructureType	3028
6.679.3.3	looseMarshal	3029
6.679.3.4	looseUnmarshal	3029
6.679.3.5	tightMarshal1	3029
6.679.3.6	tightMarshal2	3030
6.679.3.7	tightUnmarshal	3030
6.680	activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller	
	Class Reference	3031
6.680.1	Detailed Description	3032
6.680.2	Constructor & Destructor Documentation	3032
	6.680.2.1 RemoveSubscriptionInfoMarshaller	3032
	6.680.2.2 ~RemoveSubscriptionInfoMarshaller	3032
6.680.3	Member Function Documentation	3032
	6.680.3.1 createObject	3032
	6.680.3.2 getDataStructureType	3032
	6.680.3.3 looseMarshal	3033
	6.680.3.4 looseUnmarshal	3033
	6.680.3.5 tightMarshal1	3033
	6.680.3.6 tightMarshal2	3034
	6.680.3.7 tightUnmarshal	3034
6.681	activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	
	Class Reference	3035
6.681.1	Detailed Description	3036
6.681.2	Constructor & Destructor Documentation	3036
	6.681.2.1 RemoveSubscriptionInfoMarshaller	3036
	6.681.2.2 ~RemoveSubscriptionInfoMarshaller	3036
6.681.3	Member Function Documentation	3036
	6.681.3.1 createObject	3036
	6.681.3.2 getDataStructureType	3036
	6.681.3.3 looseMarshal	3037
	6.681.3.4 looseUnmarshal	3037
	6.681.3.5 tightMarshal1	3037
	6.681.3.6 tightMarshal2	3038
	6.681.3.7 tightUnmarshal	3038
6.682	activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller	
	Class Reference	3039

6.682.1 Detailed Description	3040
6.682.2 Constructor & Destructor Documentation	3040
6.682.2.1 RemoveSubscriptionInfoMarshaller	3040
6.682.2.2 ~RemoveSubscriptionInfoMarshaller	3040
6.682.3 Member Function Documentation	3040
6.682.3.1 createObject	3040
6.682.3.2 getDataStructureType	3040
6.682.3.3 looseMarshal	3041
6.682.3.4 looseUnmarshal	3041
6.682.3.5 tightMarshal1	3041
6.682.3.6 tightMarshal2	3042
6.682.3.7 tightUnmarshal	3042
6.683activemq::commands::ReplayCommand Class Reference	3043
6.683.1 Constructor & Destructor Documentation	3044
6.683.1.1 ReplayCommand	3044
6.683.1.2 ~ReplayCommand	3044
6.683.2 Member Function Documentation	3044
6.683.2.1 cloneDataStructure	3044
6.683.2.2 copyDataStructure	3044
6.683.2.3 equals	3044
6.683.2.4 getDataStructureType	3045
6.683.2.5 getFirstNakNumber	3045
6.683.2.6 getLastNakNumber	3045
6.683.2.7 setFirstNakNumber	3045
6.683.2.8 setLastNakNumber	3045
6.683.2.9 toString	3045
6.683.2.10visit	3045
6.683.3 Field Documentation	3046
6.683.3.1 firstNakNumber	3046
6.683.3.2 ID_REPLAYCOMMAND	3046
6.683.3.3 lastNakNumber	3046
6.684activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller Class Reference	3046
6.684.1 Detailed Description	3047
6.684.2 Constructor & Destructor Documentation	3047
6.684.2.1 ReplayCommandMarshaller	3047
6.684.2.2 ~ReplayCommandMarshaller	3047

6.684.3 Member Function Documentation	3047
6.684.3.1 createObject	3047
6.684.3.2 getDataStructureType	3047
6.684.3.3 looseMarshal	3048
6.684.3.4 looseUnmarshal	3048
6.684.3.5 tightMarshal1	3048
6.684.3.6 tightMarshal2	3049
6.684.3.7 tightUnmarshal	3049
6.685activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller Class	
Reference	3050
6.685.1 Detailed Description	3051
6.685.2 Constructor & Destructor Documentation	3051
6.685.2.1 ReplayCommandMarshaller	3051
6.685.2.2 ~ReplayCommandMarshaller	3051
6.685.3 Member Function Documentation	3051
6.685.3.1 createObject	3051
6.685.3.2 getDataStructureType	3051
6.685.3.3 looseMarshal	3052
6.685.3.4 looseUnmarshal	3052
6.685.3.5 tightMarshal1	3052
6.685.3.6 tightMarshal2	3053
6.685.3.7 tightUnmarshal	3053
6.686activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller Class	
Reference	3054
6.686.1 Detailed Description	3055
6.686.2 Constructor & Destructor Documentation	3055
6.686.2.1 ReplayCommandMarshaller	3055
6.686.2.2 ~ReplayCommandMarshaller	3055
6.686.3 Member Function Documentation	3055
6.686.3.1 createObject	3055
6.686.3.2 getDataStructureType	3055
6.686.3.3 looseMarshal	3056
6.686.3.4 looseUnmarshal	3056
6.686.3.5 tightMarshal1	3056
6.686.3.6 tightMarshal2	3057
6.686.3.7 tightUnmarshal	3057

6.687	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	Class	
	Reference		3058
6.687.1	Detailed Description		3059
6.687.2	Constructor & Destructor Documentation		3059
	6.687.2.1 ReplayCommandMarshaller		3059
	6.687.2.2 ~ReplayCommandMarshaller		3059
6.687.3	Member Function Documentation		3059
	6.687.3.1 createObject		3059
	6.687.3.2 getDataStructureType		3059
	6.687.3.3 looseMarshal		3060
	6.687.3.4 looseUnmarshal		3060
	6.687.3.5 tightMarshal1		3060
	6.687.3.6 tightMarshal2		3061
	6.687.3.7 tightUnmarshal		3061
6.688	activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller	Class	
	Reference		3062
6.688.1	Detailed Description		3063
6.688.2	Constructor & Destructor Documentation		3063
	6.688.2.1 ReplayCommandMarshaller		3063
	6.688.2.2 ~ReplayCommandMarshaller		3063
6.688.3	Member Function Documentation		3063
	6.688.3.1 createObject		3063
	6.688.3.2 getDataStructureType		3063
	6.688.3.3 looseMarshal		3064
	6.688.3.4 looseUnmarshal		3064
	6.688.3.5 tightMarshal1		3064
	6.688.3.6 tightMarshal2		3065
	6.688.3.7 tightUnmarshal		3065
6.689	activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller	Class	
	Reference		3066
6.689.1	Detailed Description		3067
6.689.2	Constructor & Destructor Documentation		3067
	6.689.2.1 ReplayCommandMarshaller		3067
	6.689.2.2 ~ReplayCommandMarshaller		3067
6.689.3	Member Function Documentation		3067
	6.689.3.1 createObject		3067
	6.689.3.2 getDataStructureType		3067

6.689.3.3 looseMarshal	3068
6.689.3.4 looseUnmarshal	3068
6.689.3.5 tightMarshal1	3068
6.689.3.6 tightMarshal2	3069
6.689.3.7 tightUnmarshal	3069
6.690activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference	3070
6.690.1 Constructor & Destructor Documentation	3071
6.690.1.1 ResolveProducerExecutor	3071
6.690.1.2 ~ResolveProducerExecutor	3071
6.690.2 Member Function Documentation	3071
6.690.2.1 getDestination	3071
6.690.2.2 operator=	3071
6.691activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference	3071
6.691.1 Constructor & Destructor Documentation	3072
6.691.1.1 ResolveReceiveExecutor	3072
6.691.1.2 ~ResolveReceiveExecutor	3072
6.691.2 Member Function Documentation	3072
6.691.2.1 getDestination	3072
6.691.2.2 operator=	3072
6.692decaf::internal::util::Resource Class Reference	3072
6.692.1 Detailed Description	3072
6.692.2 Constructor & Destructor Documentation	3073
6.692.2.1 ~Resource	3073
6.693decaf::internal::util::ResourceLifecycleManager Class Reference	3073
6.693.1 Detailed Description	3073
6.693.2 Constructor & Destructor Documentation	3073
6.693.2.1 ResourceLifecycleManager	3073
6.693.2.2 ~ResourceLifecycleManager	3073
6.693.3 Member Function Documentation	3073
6.693.3.1 addResource	3073
6.693.3.2 destroyResources	3073
6.694activemq::cmsutil::ResourceLifecycleManager Class Reference	3074
6.694.1 Detailed Description	3074
6.694.2 Constructor & Destructor Documentation	3075
6.694.2.1 ResourceLifecycleManager	3075
6.694.2.2 ResourceLifecycleManager	3075

6.694.2.3 ~ResourceLifecycleManager	3075
6.694.3 Member Function Documentation	3075
6.694.3.1 addConnection	3075
6.694.3.2 addDestination	3075
6.694.3.3 addMessageConsumer	3075
6.694.3.4 addMessageProducer	3075
6.694.3.5 addSession	3076
6.694.3.6 destroy	3076
6.694.3.7 operator=	3076
6.694.3.8 releaseAll	3076
6.695activemq::commands::Response Class Reference	3076
6.695.1 Constructor & Destructor Documentation	3077
6.695.1.1 Response	3077
6.695.1.2 ~Response	3077
6.695.2 Member Function Documentation	3077
6.695.2.1 cloneDataStructure	3077
6.695.2.2 copyDataStructure	3078
6.695.2.3 equals	3078
6.695.2.4 getCorrelationId	3078
6.695.2.5 getDataStructureType	3078
6.695.2.6 isResponse	3078
6.695.2.7 setCorrelationId	3079
6.695.2.8 toString	3079
6.695.2.9 visit	3079
6.695.3 Field Documentation	3079
6.695.3.1 correlationId	3079
6.695.3.2 ID_RESPONSE	3079
6.696activemq::transport::mock::ResponseBuilder Class Reference	3079
6.696.1 Detailed Description	3080
6.696.2 Constructor & Destructor Documentation	3080
6.696.2.1 ~ResponseBuilder	3080
6.696.3 Member Function Documentation	3080
6.696.3.1 buildIncomingCommands	3080
6.696.3.2 buildResponse	3081
6.697activemq::transport::correlator::ResponseCorrelator Class Reference	3081
6.697.1 Detailed Description	3082

6.697.2 Constructor & Destructor Documentation	3082
6.697.2.1 ResponseCorrelator	3082
6.697.2.2 ~ResponseCorrelator	3082
6.697.3 Member Function Documentation	3082
6.697.3.1 close	3082
6.697.3.2 onCommand	3083
6.697.3.3 oneway	3083
6.697.3.4 onTransportException	3083
6.697.3.5 request	3083
6.697.3.6 request	3084
6.697.3.7 start	3084
6.698activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class Reference .	3085
6.698.1 Detailed Description	3085
6.698.2 Constructor & Destructor Documentation	3086
6.698.2.1 ResponseMarshaller	3086
6.698.2.2 ~ResponseMarshaller	3086
6.698.3 Member Function Documentation	3086
6.698.3.1 createObject	3086
6.698.3.2 getDataStructureType	3086
6.698.3.3 looseMarshal	3086
6.698.3.4 looseUnmarshal	3087
6.698.3.5 tightMarshal1	3087
6.698.3.6 tightMarshal2	3088
6.698.3.7 tightUnmarshal	3088
6.699activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference .	3089
6.699.1 Detailed Description	3090
6.699.2 Constructor & Destructor Documentation	3090
6.699.2.1 ResponseMarshaller	3090
6.699.2.2 ~ResponseMarshaller	3090
6.699.3 Member Function Documentation	3090
6.699.3.1 createObject	3090
6.699.3.2 getDataStructureType	3091
6.699.3.3 looseMarshal	3091
6.699.3.4 looseUnmarshal	3091
6.699.3.5 tightMarshal1	3092
6.699.3.6 tightMarshal2	3092

6.699.3.7 tightUnmarshal	3093
6.700activemq::wireformat::openwire::marshal::v5::ResponseMarshaller Class Reference	3093
6.700.1 Detailed Description	3094
6.700.2 Constructor & Destructor Documentation	3095
6.700.2.1 ResponseMarshaller	3095
6.700.2.2 ~ResponseMarshaller	3095
6.700.3 Member Function Documentation	3095
6.700.3.1 createObject	3095
6.700.3.2 getDataStructureType	3095
6.700.3.3 looseMarshal	3095
6.700.3.4 looseUnmarshal	3096
6.700.3.5 tightMarshal1	3096
6.700.3.6 tightMarshal2	3097
6.700.3.7 tightUnmarshal	3097
6.701activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference	3098
6.701.1 Detailed Description	3099
6.701.2 Constructor & Destructor Documentation	3099
6.701.2.1 ResponseMarshaller	3099
6.701.2.2 ~ResponseMarshaller	3099
6.701.3 Member Function Documentation	3099
6.701.3.1 createObject	3099
6.701.3.2 getDataStructureType	3100
6.701.3.3 looseMarshal	3100
6.701.3.4 looseUnmarshal	3100
6.701.3.5 tightMarshal1	3101
6.701.3.6 tightMarshal2	3101
6.701.3.7 tightUnmarshal	3102
6.702activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class Reference	3102
6.702.1 Detailed Description	3103
6.702.2 Constructor & Destructor Documentation	3104
6.702.2.1 ResponseMarshaller	3104
6.702.2.2 ~ResponseMarshaller	3104
6.702.3 Member Function Documentation	3104
6.702.3.1 createObject	3104
6.702.3.2 getDataStructureType	3104
6.702.3.3 looseMarshal	3104

6.702.3.4 looseUnmarshal	3105
6.702.3.5 tightMarshal1	3105
6.702.3.6 tightMarshal2	3106
6.702.3.7 tightUnmarshal	3106
6.703activemq::wireformat::openwire::marshal::v6::ResponseMarshaller Class Reference .	3107
6.703.1 Detailed Description	3108
6.703.2 Constructor & Destructor Documentation	3108
6.703.2.1 ResponseMarshaller	3108
6.703.2.2 ~ResponseMarshaller	3108
6.703.3 Member Function Documentation	3108
6.703.3.1 createObject	3108
6.703.3.2 getDataStructureType	3109
6.703.3.3 looseMarshal	3109
6.703.3.4 looseUnmarshal	3109
6.703.3.5 tightMarshal1	3110
6.703.3.6 tightMarshal2	3110
6.703.3.7 tightUnmarshal	3111
6.704decaf::lang::Runnable Class Reference	3111
6.704.1 Detailed Description	3112
6.704.2 Constructor & Destructor Documentation	3112
6.704.2.1 ~Runnable	3112
6.704.3 Member Function Documentation	3112
6.704.3.1 run	3112
6.705decaf::lang::Runtime Class Reference	3112
6.705.1 Constructor & Destructor Documentation	3113
6.705.1.1 ~Runtime	3113
6.705.2 Member Function Documentation	3113
6.705.2.1 getRuntime	3113
6.705.2.2 initializeRuntime	3113
6.705.2.3 initializeRuntime	3113
6.705.2.4 shutdownRuntime	3114
6.706decaf::lang::exceptions::RuntimeException Class Reference	3114
6.706.1 Constructor & Destructor Documentation	3115
6.706.1.1 RuntimeException	3115
6.706.1.2 RuntimeException	3115
6.706.1.3 RuntimeException	3115

6.706.1.4 RuntimeException	3115
6.706.1.5 RuntimeException	3115
6.706.1.6 RuntimeException	3116
6.706.1.7 ~RuntimeException	3116
6.706.2 Member Function Documentation	3116
6.706.2.1 clone	3116
6.707decaf::security::SecureRandom Class Reference	3116
6.707.1 Detailed Description	3118
6.707.2 Constructor & Destructor Documentation	3118
6.707.2.1 SecureRandom	3118
6.707.2.2 SecureRandom	3118
6.707.2.3 SecureRandom	3118
6.707.2.4 ~SecureRandom	3119
6.707.3 Member Function Documentation	3119
6.707.3.1 next	3119
6.707.3.2 nextBytes	3119
6.707.3.3 nextBytes	3120
6.707.3.4 setSeed	3120
6.707.3.5 setSeed	3120
6.707.3.6 setSeed	3121
6.708decaf::internal::security::SecureRandomImpl Class Reference	3121
6.708.1 Detailed Description	3122
6.708.2 Constructor & Destructor Documentation	3122
6.708.2.1 SecureRandomImpl	3122
6.708.2.2 ~SecureRandomImpl	3122
6.708.2.3 SecureRandomImpl	3122
6.708.2.4 ~SecureRandomImpl	3122
6.708.3 Member Function Documentation	3122
6.708.3.1 providerGenerateSeed	3122
6.708.3.2 providerGenerateSeed	3123
6.708.3.3 providerNextBytes	3123
6.708.3.4 providerNextBytes	3123
6.708.3.5 providerSetSeed	3124
6.708.3.6 providerSetSeed	3124
6.709decaf::security::SecureRandomSpi Class Reference	3124
6.709.1 Detailed Description	3125

6.709.2 Constructor & Destructor Documentation	3125
6.709.2.1 SecureRandomSpi	3125
6.709.2.2 ~SecureRandomSpi	3125
6.709.3 Member Function Documentation	3125
6.709.3.1 providerGenerateSeed	3125
6.709.3.2 providerNextBytes	3125
6.709.3.3 providerSetSeed	3126
6.710decaf::util::concurrent::Semaphore Class Reference	3126
6.710.1 Detailed Description	3127
6.710.2 Constructor & Destructor Documentation	3129
6.710.2.1 Semaphore	3129
6.710.2.2 Semaphore	3129
6.710.2.3 ~Semaphore	3129
6.710.3 Member Function Documentation	3129
6.710.3.1 acquire	3129
6.710.3.2 acquire	3130
6.710.3.3 acquireUninterruptibly	3130
6.710.3.4 acquireUninterruptibly	3131
6.710.3.5 availablePermits	3131
6.710.3.6 drainPermits	3131
6.710.3.7 isFair	3132
6.710.3.8 release	3132
6.710.3.9 release	3132
6.710.3.10 toString	3132
6.710.3.11 tryAcquire	3133
6.710.3.12 tryAcquire	3133
6.710.3.13 tryAcquire	3134
6.710.3.14 tryAcquire	3135
6.711activemq::cmsutil::CmsTemplate::SendExecutor Class Reference	3135
6.711.1 Constructor & Destructor Documentation	3136
6.711.1.1 SendExecutor	3136
6.711.1.2 SendExecutor	3136
6.711.1.3 ~SendExecutor	3136
6.711.2 Member Function Documentation	3136
6.711.2.1 doInCms	3136
6.711.2.2 operator=	3136

6.712	decaf::net::ServerSocket Class Reference	3136
6.712.1	Detailed Description	3138
6.712.2	Constructor & Destructor Documentation	3139
6.712.2.1	ServerSocket	3139
6.712.2.2	ServerSocket	3139
6.712.2.3	ServerSocket	3139
6.712.2.4	ServerSocket	3140
6.712.2.5	~ServerSocket	3140
6.712.2.6	ServerSocket	3140
6.712.3	Member Function Documentation	3141
6.712.3.1	accept	3141
6.712.3.2	bind	3141
6.712.3.3	bind	3141
6.712.3.4	checkClosed	3142
6.712.3.5	close	3142
6.712.3.6	ensureCreated	3142
6.712.3.7	getDefaultBacklog	3142
6.712.3.8	getLocalPort	3142
6.712.3.9	getReceiveBufferSize	3142
6.712.3.10	getReuseAddress	3143
6.712.3.11	getSoTimeout	3143
6.712.3.12	implAccept	3143
6.712.3.13	sBound	3143
6.712.3.14	sClosed	3144
6.712.3.15	setReceiveBufferSize	3144
6.712.3.16	setReuseAddress	3144
6.712.3.17	setSocketImplFactory	3144
6.712.3.18	setSoTimeout	3145
6.712.3.19	setUpSocketImpl	3145
6.712.3.20	toString	3145
6.713	decaf::net::ServerSocketFactory Class Reference	3145
6.713.1	Detailed Description	3146
6.713.2	Constructor & Destructor Documentation	3146
6.713.2.1	ServerSocketFactory	3146
6.713.2.2	~ServerSocketFactory	3146
6.713.3	Member Function Documentation	3146

6.713.3.1 createServerSocket	3146
6.713.3.2 createServerSocket	3147
6.713.3.3 createServerSocket	3147
6.713.3.4 createServerSocket	3147
6.713.3.5 getDefault	3148
6.714cms::Session Class Reference	3148
6.714.1 Detailed Description	3150
6.714.2 Member Enumeration Documentation	3151
6.714.2.1 AcknowledgeMode	3151
6.714.3 Constructor & Destructor Documentation	3152
6.714.3.1 ~Session	3152
6.714.4 Member Function Documentation	3152
6.714.4.1 close	3152
6.714.4.2 commit	3152
6.714.4.3 createBrowser	3152
6.714.4.4 createBrowser	3153
6.714.4.5 createBytesMessage	3153
6.714.4.6 createBytesMessage	3153
6.714.4.7 createConsumer	3154
6.714.4.8 createConsumer	3154
6.714.4.9 createConsumer	3155
6.714.4.10 createDurableConsumer	3155
6.714.4.11 createMapMessage	3156
6.714.4.12 createMessage	3156
6.714.4.13 createProducer	3156
6.714.4.14 createQueue	3156
6.714.4.15 createStreamMessage	3157
6.714.4.16 createTemporaryQueue	3157
6.714.4.17 createTemporaryTopic	3157
6.714.4.18 createTextMessage	3158
6.714.4.19 createTextMessage	3158
6.714.4.20 createTopic	3158
6.714.4.21 getAcknowledgeMode	3158
6.714.4.22 isTransacted	3159
6.714.4.23 recover	3159
6.714.4.24 rollback	3159

6.714.4.25	unsubscribe	3160
6.715	activemq::cmsutil::SessionCallback Class Reference	3160
6.715.1	Detailed Description	3160
6.715.2	Constructor & Destructor Documentation	3161
6.715.2.1	~SessionCallback	3161
6.715.3	Member Function Documentation	3161
6.715.3.1	doInCms	3161
6.716	activemq::commands::SessionId Class Reference	3161
6.716.1	Member Typedef Documentation	3163
6.716.1.1	COMPARATOR	3163
6.716.2	Constructor & Destructor Documentation	3163
6.716.2.1	SessionId	3163
6.716.2.2	SessionId	3163
6.716.2.3	SessionId	3163
6.716.2.4	SessionId	3163
6.716.2.5	SessionId	3163
6.716.2.6	~SessionId	3163
6.716.3	Member Function Documentation	3163
6.716.3.1	cloneDataStructure	3163
6.716.3.2	compareTo	3163
6.716.3.3	copyDataStructure	3163
6.716.3.4	equals	3163
6.716.3.5	equals	3164
6.716.3.6	getConnectionId	3164
6.716.3.7	getConnectionId	3164
6.716.3.8	getDataStructureType	3164
6.716.3.9	getParentId	3164
6.716.3.10	getValue	3164
6.716.3.11	operator<	3164
6.716.3.12	operator=	3164
6.716.3.13	operator==	3164
6.716.3.14	setConnectionId	3164
6.716.3.15	setValue	3164
6.716.3.16	toString	3164
6.716.4	Field Documentation	3165
6.716.4.1	connectionId	3165

6.716.4.2 ID_SESSIONID	3165
6.716.4.3 value	3165
6.717activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference	3165
6.717.1 Detailed Description	3166
6.717.2 Constructor & Destructor Documentation	3166
6.717.2.1 SessionIdMarshaller	3166
6.717.2.2 ~SessionIdMarshaller	3166
6.717.3 Member Function Documentation	3166
6.717.3.1 createObject	3166
6.717.3.2 getDataStructureType	3166
6.717.3.3 looseMarshal	3167
6.717.3.4 looseUnmarshal	3167
6.717.3.5 tightMarshal1	3167
6.717.3.6 tightMarshal2	3168
6.717.3.7 tightUnmarshal	3168
6.718activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference	3169
6.718.1 Detailed Description	3170
6.718.2 Constructor & Destructor Documentation	3170
6.718.2.1 SessionIdMarshaller	3170
6.718.2.2 ~SessionIdMarshaller	3170
6.718.3 Member Function Documentation	3170
6.718.3.1 createObject	3170
6.718.3.2 getDataStructureType	3170
6.718.3.3 looseMarshal	3171
6.718.3.4 looseUnmarshal	3171
6.718.3.5 tightMarshal1	3171
6.718.3.6 tightMarshal2	3172
6.718.3.7 tightUnmarshal	3172
6.719activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller Class Reference	3173
6.719.1 Detailed Description	3174
6.719.2 Constructor & Destructor Documentation	3174
6.719.2.1 SessionIdMarshaller	3174
6.719.2.2 ~SessionIdMarshaller	3174
6.719.3 Member Function Documentation	3174
6.719.3.1 createObject	3174
6.719.3.2 getDataStructureType	3174

6.719.3.3 looseMarshal	3174
6.719.3.4 looseUnmarshal	3175
6.719.3.5 tightMarshal1	3175
6.719.3.6 tightMarshal2	3176
6.719.3.7 tightUnmarshal	3176
6.720activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference .	3176
6.720.1 Detailed Description	3177
6.720.2 Constructor & Destructor Documentation	3178
6.720.2.1 SessionIdMarshaller	3178
6.720.2.2 ~SessionIdMarshaller	3178
6.720.3 Member Function Documentation	3178
6.720.3.1 createObject	3178
6.720.3.2 getDataStructureType	3178
6.720.3.3 looseMarshal	3178
6.720.3.4 looseUnmarshal	3179
6.720.3.5 tightMarshal1	3179
6.720.3.6 tightMarshal2	3179
6.720.3.7 tightUnmarshal	3180
6.721activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference .	3180
6.721.1 Detailed Description	3181
6.721.2 Constructor & Destructor Documentation	3182
6.721.2.1 SessionIdMarshaller	3182
6.721.2.2 ~SessionIdMarshaller	3182
6.721.3 Member Function Documentation	3182
6.721.3.1 createObject	3182
6.721.3.2 getDataStructureType	3182
6.721.3.3 looseMarshal	3182
6.721.3.4 looseUnmarshal	3183
6.721.3.5 tightMarshal1	3183
6.721.3.6 tightMarshal2	3183
6.721.3.7 tightUnmarshal	3184
6.722activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class Reference .	3184
6.722.1 Detailed Description	3185
6.722.2 Constructor & Destructor Documentation	3186
6.722.2.1 SessionIdMarshaller	3186
6.722.2.2 ~SessionIdMarshaller	3186

6.722.3 Member Function Documentation	3186
6.722.3.1 createObject	3186
6.722.3.2 getDataStructureType	3186
6.722.3.3 looseMarshal	3186
6.722.3.4 looseUnmarshal	3187
6.722.3.5 tightMarshal1	3187
6.722.3.6 tightMarshal2	3187
6.722.3.7 tightUnmarshal	3188
6.723activemq::commands::SessionInfo Class Reference	3188
6.723.1 Constructor & Destructor Documentation	3189
6.723.1.1 SessionInfo	3189
6.723.1.2 ~SessionInfo	3189
6.723.2 Member Function Documentation	3189
6.723.2.1 cloneDataStructure	3189
6.723.2.2 copyDataStructure	3190
6.723.2.3 createRemoveCommand	3190
6.723.2.4 equals	3190
6.723.2.5 getAckMode	3190
6.723.2.6 getDataStructureType	3190
6.723.2.7 getSessionId	3191
6.723.2.8 getSessionId	3191
6.723.2.9 setAckMode	3191
6.723.2.10setSessionId	3191
6.723.2.11toString	3191
6.723.2.12visit	3191
6.723.3 Field Documentation	3191
6.723.3.1 ID_SESSIONINFO	3191
6.723.3.2 sessionId	3191
6.724activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller Class Reference	3192
6.724.1 Detailed Description	3192
6.724.2 Constructor & Destructor Documentation	3193
6.724.2.1 SessionInfoMarshaller	3193
6.724.2.2 ~SessionInfoMarshaller	3193
6.724.3 Member Function Documentation	3193
6.724.3.1 createObject	3193
6.724.3.2 getDataStructureType	3193

6.724.3.3 looseMarshal	3193
6.724.3.4 looseUnmarshal	3194
6.724.3.5 tightMarshal1	3194
6.724.3.6 tightMarshal2	3194
6.724.3.7 tightUnmarshal	3195
6.725activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller Class Reference	3195
6.725.1 Detailed Description	3196
6.725.2 Constructor & Destructor Documentation	3197
6.725.2.1 SessionInfoMarshaller	3197
6.725.2.2 ~SessionInfoMarshaller	3197
6.725.3 Member Function Documentation	3197
6.725.3.1 createObject	3197
6.725.3.2 getDataStructureType	3197
6.725.3.3 looseMarshal	3197
6.725.3.4 looseUnmarshal	3198
6.725.3.5 tightMarshal1	3198
6.725.3.6 tightMarshal2	3198
6.725.3.7 tightUnmarshal	3199
6.726activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller Class Reference	3199
6.726.1 Detailed Description	3200
6.726.2 Constructor & Destructor Documentation	3201
6.726.2.1 SessionInfoMarshaller	3201
6.726.2.2 ~SessionInfoMarshaller	3201
6.726.3 Member Function Documentation	3201
6.726.3.1 createObject	3201
6.726.3.2 getDataStructureType	3201
6.726.3.3 looseMarshal	3201
6.726.3.4 looseUnmarshal	3202
6.726.3.5 tightMarshal1	3202
6.726.3.6 tightMarshal2	3202
6.726.3.7 tightUnmarshal	3203
6.727activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller Class Reference	3203
6.727.1 Detailed Description	3204
6.727.2 Constructor & Destructor Documentation	3205
6.727.2.1 SessionInfoMarshaller	3205
6.727.2.2 ~SessionInfoMarshaller	3205

6.727.3 Member Function Documentation	3205
6.727.3.1 createObject	3205
6.727.3.2 getDataStructureType	3205
6.727.3.3 looseMarshal	3205
6.727.3.4 looseUnmarshal	3206
6.727.3.5 tightMarshal1	3206
6.727.3.6 tightMarshal2	3206
6.727.3.7 tightUnmarshal	3207
6.728activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller Class Reference	3207
6.728.1 Detailed Description	3208
6.728.2 Constructor & Destructor Documentation	3209
6.728.2.1 SessionInfoMarshaller	3209
6.728.2.2 ~SessionInfoMarshaller	3209
6.728.3 Member Function Documentation	3209
6.728.3.1 createObject	3209
6.728.3.2 getDataStructureType	3209
6.728.3.3 looseMarshal	3209
6.728.3.4 looseUnmarshal	3210
6.728.3.5 tightMarshal1	3210
6.728.3.6 tightMarshal2	3210
6.728.3.7 tightUnmarshal	3211
6.729activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller Class Reference	3211
6.729.1 Detailed Description	3212
6.729.2 Constructor & Destructor Documentation	3213
6.729.2.1 SessionInfoMarshaller	3213
6.729.2.2 ~SessionInfoMarshaller	3213
6.729.3 Member Function Documentation	3213
6.729.3.1 createObject	3213
6.729.3.2 getDataStructureType	3213
6.729.3.3 looseMarshal	3213
6.729.3.4 looseUnmarshal	3214
6.729.3.5 tightMarshal1	3214
6.729.3.6 tightMarshal2	3214
6.729.3.7 tightUnmarshal	3215
6.730activemq::cmsutil::SessionPool Class Reference	3215
6.730.1 Detailed Description	3216

6.730.2 Constructor & Destructor Documentation	3216
6.730.2.1 SessionPool	3216
6.730.2.2 SessionPool	3216
6.730.2.3 ~SessionPool	3216
6.730.3 Member Function Documentation	3217
6.730.3.1 getResourceLifecycleManager	3217
6.730.3.2 operator=	3217
6.730.3.3 returnSession	3217
6.730.3.4 takeSession	3217
6.731activemq::state::SessionState Class Reference	3217
6.731.1 Constructor & Destructor Documentation	3219
6.731.1.1 SessionState	3219
6.731.1.2 ~SessionState	3219
6.731.2 Member Function Documentation	3219
6.731.2.1 addConsumer	3219
6.731.2.2 addProducer	3219
6.731.2.3 checkShutdown	3219
6.731.2.4 getConsumerState	3219
6.731.2.5 getConsumerStates	3219
6.731.2.6 getInfo	3219
6.731.2.7 getProducerState	3219
6.731.2.8 getProducerStates	3219
6.731.2.9 removeConsumer	3219
6.731.2.10removeProducer	3219
6.731.2.11shutdown	3219
6.731.2.12toString	3219
6.732decaf::util::Set< E > Class Template Reference	3220
6.732.1 Detailed Description	3220
6.732.2 Constructor & Destructor Documentation	3220
6.732.2.1 ~Set	3220
6.733decaf::lang::Short Class Reference	3220
6.733.1 Constructor & Destructor Documentation	3222
6.733.1.1 Short	3222
6.733.1.2 Short	3223
6.733.1.3 ~Short	3223
6.733.2 Member Function Documentation	3223

6.733.2.1	byteValue	3223
6.733.2.2	compareTo	3223
6.733.2.3	compareTo	3223
6.733.2.4	decode	3224
6.733.2.5	doubleValue	3224
6.733.2.6	equals	3224
6.733.2.7	equals	3224
6.733.2.8	float Value	3224
6.733.2.9	int Value	3225
6.733.2.10	longValue	3225
6.733.2.11	operator<	3225
6.733.2.12	operator<	3225
6.733.2.13	operator==	3226
6.733.2.14	operator==	3226
6.733.2.15	parseShort	3226
6.733.2.16	parseShort	3227
6.733.2.17	reverseBytes	3227
6.733.2.18	short Value	3227
6.733.2.19	toString	3227
6.733.2.20	toString	3228
6.733.2.21	valueOf	3228
6.733.2.22	valueOf	3228
6.733.2.23	valueOf	3228
6.733.3	Field Documentation	3229
6.733.3.1	MAX_VALUE	3229
6.733.3.2	MIN_VALUE	3229
6.733.3.3	SIZE	3229
6.734	decaf::internal::nio::ShortArrayBuffer Class Reference	3229
6.734.1	Constructor & Destructor Documentation	3232
6.734.1.1	ShortArrayBuffer	3232
6.734.1.2	ShortArrayBuffer	3233
6.734.1.3	ShortArrayBuffer	3233
6.734.1.4	ShortArrayBuffer	3234
6.734.1.5	~ShortArrayBuffer	3234
6.734.2	Member Function Documentation	3234
6.734.2.1	array	3234

6.734.2.2 arrayOffset	3234
6.734.2.3 asReadOnlyBuffer	3235
6.734.2.4 compact	3235
6.734.2.5 duplicate	3235
6.734.2.6 get	3236
6.734.2.7 get	3236
6.734.2.8 hasArray	3237
6.734.2.9 isReadOnly	3237
6.734.2.10put	3237
6.734.2.11put	3237
6.734.2.12setReadOnly	3238
6.734.2.13slice	3238
6.735decaf::nio::ShortBuffer Class Reference	3238
6.735.1 Detailed Description	3241
6.735.2 Constructor & Destructor Documentation	3241
6.735.2.1 ShortBuffer	3241
6.735.2.2 ~ShortBuffer	3241
6.735.3 Member Function Documentation	3241
6.735.3.1 allocate	3241
6.735.3.2 array	3242
6.735.3.3 arrayOffset	3242
6.735.3.4 asReadOnlyBuffer	3242
6.735.3.5 compact	3243
6.735.3.6 compareTo	3243
6.735.3.7 duplicate	3243
6.735.3.8 equals	3244
6.735.3.9 get	3244
6.735.3.10get	3244
6.735.3.11get	3244
6.735.3.12get	3245
6.735.3.13hasArray	3245
6.735.3.14operator<	3246
6.735.3.15operator==	3246
6.735.3.16put	3246
6.735.3.17put	3246
6.735.3.18put	3246

6.735.3.1	put	3247
6.735.3.2	put	3247
6.735.3.2	slice	3248
6.735.3.2	toString	3248
6.735.3.2	wrap	3248
6.735.3.2	wrap	3249
6.736	activemq::commands::ShutdownInfo Class Reference	3249
6.736.1	Constructor & Destructor Documentation	3250
6.736.1.1	ShutdownInfo	3250
6.736.1.2	~ShutdownInfo	3250
6.736.2	Member Function Documentation	3250
6.736.2.1	cloneDataStructure	3250
6.736.2.2	copyDataStructure	3251
6.736.2.3	equals	3251
6.736.2.4	getDataStructureType	3251
6.736.2.5	isShutdownInfo	3251
6.736.2.6	toString	3251
6.736.2.7	visit	3252
6.736.3	Field Documentation	3252
6.736.3.1	ID_SHUTDOWNINFO	3252
6.737	activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller Class Refer- ence	3252
6.737.1	Detailed Description	3253
6.737.2	Constructor & Destructor Documentation	3253
6.737.2.1	ShutdownInfoMarshaller	3253
6.737.2.2	~ShutdownInfoMarshaller	3253
6.737.3	Member Function Documentation	3253
6.737.3.1	createObject	3253
6.737.3.2	getDataStructureType	3254
6.737.3.3	looseMarshal	3254
6.737.3.4	looseUnmarshal	3254
6.737.3.5	tightMarshal1	3255
6.737.3.6	tightMarshal2	3255
6.737.3.7	tightUnmarshal	3255
6.738	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller Class Refer- ence	3256
6.738.1	Detailed Description	3257

6.738.2 Constructor & Destructor Documentation	3257
6.738.2.1 ShutdownInfoMarshaller	3257
6.738.2.2 ~ShutdownInfoMarshaller	3257
6.738.3 Member Function Documentation	3257
6.738.3.1 createObject	3257
6.738.3.2 getDataStructureType	3257
6.738.3.3 looseMarshal	3258
6.738.3.4 looseUnmarshal	3258
6.738.3.5 tightMarshal1	3258
6.738.3.6 tightMarshal2	3259
6.738.3.7 tightUnmarshal	3259
6.739activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller Class Refer- ence	3260
6.739.1 Detailed Description	3261
6.739.2 Constructor & Destructor Documentation	3261
6.739.2.1 ShutdownInfoMarshaller	3261
6.739.2.2 ~ShutdownInfoMarshaller	3261
6.739.3 Member Function Documentation	3261
6.739.3.1 createObject	3261
6.739.3.2 getDataStructureType	3261
6.739.3.3 looseMarshal	3262
6.739.3.4 looseUnmarshal	3262
6.739.3.5 tightMarshal1	3262
6.739.3.6 tightMarshal2	3263
6.739.3.7 tightUnmarshal	3263
6.740activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller Class Refer- ence	3264
6.740.1 Detailed Description	3265
6.740.2 Constructor & Destructor Documentation	3265
6.740.2.1 ShutdownInfoMarshaller	3265
6.740.2.2 ~ShutdownInfoMarshaller	3265
6.740.3 Member Function Documentation	3265
6.740.3.1 createObject	3265
6.740.3.2 getDataStructureType	3265
6.740.3.3 looseMarshal	3266
6.740.3.4 looseUnmarshal	3266
6.740.3.5 tightMarshal1	3266

6.740.3.6 tightMarshal2	3267
6.740.3.7 tightUnmarshal	3267
6.741activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller Class Reference	3268
6.741.1 Detailed Description	3269
6.741.2 Constructor & Destructor Documentation	3269
6.741.2.1 ShutdownInfoMarshaller	3269
6.741.2.2 ~ShutdownInfoMarshaller	3269
6.741.3 Member Function Documentation	3269
6.741.3.1 createObject	3269
6.741.3.2 getDataStructureType	3269
6.741.3.3 looseMarshal	3270
6.741.3.4 looseUnmarshal	3270
6.741.3.5 tightMarshal1	3270
6.741.3.6 tightMarshal2	3271
6.741.3.7 tightUnmarshal	3271
6.742activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller Class Reference	3272
6.742.1 Detailed Description	3273
6.742.2 Constructor & Destructor Documentation	3273
6.742.2.1 ShutdownInfoMarshaller	3273
6.742.2.2 ~ShutdownInfoMarshaller	3273
6.742.3 Member Function Documentation	3273
6.742.3.1 createObject	3273
6.742.3.2 getDataStructureType	3273
6.742.3.3 looseMarshal	3274
6.742.3.4 looseUnmarshal	3274
6.742.3.5 tightMarshal1	3274
6.742.3.6 tightMarshal2	3275
6.742.3.7 tightUnmarshal	3275
6.743decaf::security::SignatureException Class Reference	3276
6.743.1 Constructor & Destructor Documentation	3276
6.743.1.1 SignatureException	3276
6.743.1.2 SignatureException	3277
6.743.1.3 SignatureException	3277
6.743.1.4 SignatureException	3277
6.743.1.5 SignatureException	3277

6.743.1.6 SignatureException	3277
6.743.1.7 ~SignatureException	3278
6.743.2 Member Function Documentation	3278
6.743.2.1 clone	3278
6.744decaf::util::logging::SimpleFormatter Class Reference	3278
6.744.1 Detailed Description	3279
6.744.2 Constructor & Destructor Documentation	3279
6.744.2.1 SimpleFormatter	3279
6.744.2.2 ~SimpleFormatter	3279
6.744.3 Member Function Documentation	3279
6.744.3.1 format	3279
6.745decaf::util::logging::SimpleLogger Class Reference	3279
6.745.1 Constructor & Destructor Documentation	3280
6.745.1.1 SimpleLogger	3280
6.745.1.2 ~SimpleLogger	3280
6.745.2 Member Function Documentation	3280
6.745.2.1 debug	3280
6.745.2.2 error	3280
6.745.2.3 fatal	3280
6.745.2.4 info	3280
6.745.2.5 log	3281
6.745.2.6 mark	3281
6.745.2.7 warn	3281
6.746decaf::net::Socket Class Reference	3281
6.746.1 Detailed Description	3284
6.746.2 Constructor & Destructor Documentation	3285
6.746.2.1 Socket	3285
6.746.2.2 Socket	3285
6.746.2.3 Socket	3285
6.746.2.4 Socket	3285
6.746.2.5 Socket	3286
6.746.2.6 Socket	3286
6.746.2.7 ~Socket	3287
6.746.3 Member Function Documentation	3287
6.746.3.1 accepted	3287
6.746.3.2 bind	3287

6.746.3.3	checkClosed	3287
6.746.3.4	close	3287
6.746.3.5	connect	3288
6.746.3.6	connect	3288
6.746.3.7	ensureCreated	3288
6.746.3.8	getInetAddress	3288
6.746.3.9	getInputStream	3289
6.746.3.10	getKeepAlive	3289
6.746.3.11	getLocalAddress	3289
6.746.3.12	getLocalPort	3289
6.746.3.13	getOOBInline	3290
6.746.3.14	getOutputStream	3290
6.746.3.15	getPort	3290
6.746.3.16	getReceiveBufferSize	3290
6.746.3.17	getReuseAddress	3291
6.746.3.18	getSendBufferSize	3291
6.746.3.19	getSoLinger	3291
6.746.3.20	getSoTimeout	3291
6.746.3.21	getTcpNoDelay	3292
6.746.3.22	getTrafficClass	3292
6.746.3.23	initSocketImpl	3292
6.746.3.24	isBound	3292
6.746.3.25	isClosed	3292
6.746.3.26	isConnected	3293
6.746.3.27	isInputShutdown	3293
6.746.3.28	isOutputShutdown	3293
6.746.3.29	sendUrgentData	3293
6.746.3.30	setKeepAlive	3293
6.746.3.31	setOOBInline	3294
6.746.3.32	setReceiveBufferSize	3294
6.746.3.33	setReuseAddress	3294
6.746.3.34	setSendBufferSize	3294
6.746.3.35	setSocketImplFactory	3295
6.746.3.36	setSoLinger	3295
6.746.3.37	setSoTimeout	3295
6.746.3.38	setTcpNoDelay	3296

6.746.3.39	setTrafficClass	3296
6.746.3.40	shutdownInput	3296
6.746.3.41	shutdownOutput	3297
6.746.3.42	toString	3297
6.746.4	Friends And Related Function Documentation	3297
6.746.4.1	ServerSocket	3297
6.746.5	Field Documentation	3297
6.746.5.1	impl	3297
6.747	decaf::net::SocketAddress Class Reference	3297
6.747.1	Detailed Description	3297
6.747.2	Constructor & Destructor Documentation	3298
6.747.2.1	~SocketAddress	3298
6.748	decaf::net::SocketError Class Reference	3298
6.748.1	Detailed Description	3298
6.748.2	Member Function Documentation	3298
6.748.2.1	getErrorCode	3298
6.748.2.2	getErrorString	3298
6.749	decaf::net::SocketException Class Reference	3298
6.749.1	Detailed Description	3299
6.749.2	Constructor & Destructor Documentation	3299
6.749.2.1	SocketException	3299
6.749.2.2	SocketException	3299
6.749.2.3	SocketException	3299
6.749.2.4	SocketException	3299
6.749.2.5	SocketException	3300
6.749.2.6	SocketException	3300
6.749.2.7	~SocketException	3300
6.749.3	Member Function Documentation	3300
6.749.3.1	clone	3300
6.750	decaf::net::SocketFactory Class Reference	3301
6.750.1	Detailed Description	3302
6.750.2	Constructor & Destructor Documentation	3302
6.750.2.1	SocketFactory	3302
6.750.2.2	~SocketFactory	3302
6.750.3	Member Function Documentation	3302
6.750.3.1	createSocket	3302

6.750.3.2 createSocket	3302
6.750.3.3 createSocket	3303
6.750.3.4 createSocket	3303
6.750.3.5 createSocket	3304
6.750.3.6 getDefault	3304
6.751decaf::internal::net::SocketFileDescriptor Class Reference	3305
6.751.1 Detailed Description	3305
6.751.2 Constructor & Destructor Documentation	3305
6.751.2.1 SocketFileDescriptor	3305
6.751.2.2 ~SocketFileDescriptor	3305
6.751.3 Member Function Documentation	3305
6.751.3.1 getValue	3305
6.752decaf::net::SocketImpl Class Reference	3306
6.752.1 Detailed Description	3308
6.752.2 Constructor & Destructor Documentation	3308
6.752.2.1 SocketImpl	3308
6.752.2.2 ~SocketImpl	3308
6.752.3 Member Function Documentation	3308
6.752.3.1 accept	3308
6.752.3.2 available	3308
6.752.3.3 bind	3308
6.752.3.4 close	3309
6.752.3.5 connect	3309
6.752.3.6 create	3309
6.752.3.7 getFileDescriptor	3310
6.752.3.8 getInetAddress	3310
6.752.3.9 getInputStream	3310
6.752.3.10getLocalAddress	3310
6.752.3.11getLocalPort	3310
6.752.3.12getOption	3311
6.752.3.13getOutputStream	3311
6.752.3.14getPort	3311
6.752.3.15listen	3311
6.752.3.16sendUrgentData	3312
6.752.3.17setOption	3312
6.752.3.18shutdownInput	3312

6.752.3.1	shutdownOutput	3313
6.752.3.2	supportsUrgentData	3313
6.752.3.2	toString	3313
6.752.4	Field Documentation	3313
6.752.4.1	address	3313
6.752.4.2	fd	3313
6.752.4.3	localPort	3313
6.752.4.4	port	3313
6.753	decaf::net::SocketImplFactory Class Reference	3314
6.753.1	Detailed Description	3314
6.753.2	Constructor & Destructor Documentation	3314
6.753.2.1	~SocketImplFactory	3314
6.753.3	Member Function Documentation	3314
6.753.3.1	createSocketImpl	3314
6.754	decaf::net::SocketOptions Class Reference	3315
6.754.1	Detailed Description	3316
6.754.2	Constructor & Destructor Documentation	3316
6.754.2.1	~SocketOptions	3316
6.754.3	Field Documentation	3316
6.754.3.1	SOCKET_OPTION_BINDADDR	3316
6.754.3.2	SOCKET_OPTION_BROADCAST	3316
6.754.3.3	SOCKET_OPTION_IP_MULTICAST_IF	3316
6.754.3.4	SOCKET_OPTION_IP_MULTICAST_IF2	3317
6.754.3.5	SOCKET_OPTION_IP_MULTICAST_LOOP	3317
6.754.3.6	SOCKET_OPTION_IP_TOS	3317
6.754.3.7	SOCKET_OPTION_KEEPALIVE	3317
6.754.3.8	SOCKET_OPTION_LINGER	3317
6.754.3.9	SOCKET_OPTION_OOBLINE	3318
6.754.3.10	SOCKET_OPTION_RCVBUF	3318
6.754.3.11	SOCKET_OPTION_REUSEADDR	3318
6.754.3.12	SOCKET_OPTION_SNDBUF	3318
6.754.3.13	SOCKET_OPTION_TCP_NODELAY	3318
6.754.3.14	SOCKET_OPTION_TIMEOUT	3318
6.755	decaf::net::SocketTimeoutException Class Reference	3319
6.755.1	Constructor & Destructor Documentation	3319
6.755.1.1	SocketTimeoutException	3319

6.755.1.2 SocketTimeoutException	3319
6.755.1.3 SocketTimeoutException	3320
6.755.1.4 SocketTimeoutException	3320
6.755.1.5 SocketTimeoutException	3320
6.755.1.6 SocketTimeoutException	3320
6.755.1.7 ~SocketTimeoutException	3321
6.755.2 Member Function Documentation	3321
6.755.2.1 clone	3321
6.756decaf::net::ssl::SSLContext Class Reference	3321
6.756.1 Detailed Description	3322
6.756.2 Constructor & Destructor Documentation	3322
6.756.2.1 SSLContext	3322
6.756.2.2 ~SSLContext	3322
6.756.3 Member Function Documentation	3322
6.756.3.1 getDefault	3322
6.756.3.2 getDefaultSSLParameters	3322
6.756.3.3 getServerSocketFactory	3322
6.756.3.4 getSocketFactory	3323
6.756.3.5 getSupportedSSLParameters	3323
6.756.3.6 setDefault	3323
6.757decaf::net::ssl::SSLContextSpi Class Reference	3324
6.757.1 Detailed Description	3324
6.757.2 Constructor & Destructor Documentation	3324
6.757.2.1 ~SSLContextSpi	3324
6.757.3 Member Function Documentation	3324
6.757.3.1 providerGetDefaultSSLParameters	3324
6.757.3.2 providerGetServerSocketFactory	3325
6.757.3.3 providerGetSocketFactory	3325
6.757.3.4 providerGetSupportedSSLParameters	3326
6.757.3.5 providerInit	3326
6.758decaf::net::ssl::SSLParameters Class Reference	3326
6.758.1 Constructor & Destructor Documentation	3327
6.758.1.1 SSLParameters	3327
6.758.1.2 SSLParameters	3327
6.758.1.3 SSLParameters	3327
6.758.1.4 ~SSLParameters	3328

6.758.2 Member Function Documentation	3328
6.758.2.1 getCipherSuites	3328
6.758.2.2 getNeedClientAuth	3328
6.758.2.3 getProtocols	3328
6.758.2.4 getWantClientAuth	3328
6.758.2.5 setCipherSuites	3328
6.758.2.6 setNeedClientAuth	3328
6.758.2.7 setProtocols	3329
6.758.2.8 setWantClientAuth	3329
6.759decaf::net::ssl::SSLServerSocket Class Reference	3329
6.759.1 Detailed Description	3330
6.759.2 Constructor & Destructor Documentation	3331
6.759.2.1 SSLServerSocket	3331
6.759.2.2 SSLServerSocket	3331
6.759.2.3 SSLServerSocket	3331
6.759.2.4 SSLServerSocket	3332
6.759.2.5 ~SSLServerSocket	3332
6.759.3 Member Function Documentation	3332
6.759.3.1 getEnabledCipherSuites	3332
6.759.3.2 getEnabledProtocols	3332
6.759.3.3 getNeedClientAuth	3333
6.759.3.4 getSupportedCipherSuites	3333
6.759.3.5 getSupportedProtocols	3333
6.759.3.6 getWantClientAuth	3333
6.759.3.7 setEnabledCipherSuites	3333
6.759.3.8 setEnabledProtocols	3334
6.759.3.9 setNeedClientAuth	3334
6.759.3.10set WantClientAuth	3334
6.760decaf::net::ssl::SSLServerSocketFactory Class Reference	3335
6.760.1 Detailed Description	3335
6.760.2 Constructor & Destructor Documentation	3336
6.760.2.1 SSLServerSocketFactory	3336
6.760.2.2 ~SSLServerSocketFactory	3336
6.760.3 Member Function Documentation	3336
6.760.3.1 getDefault	3336
6.760.3.2 getDefaultCipherSuites	3336

6.760.3.3	getSupportedCipherSuites	3336
6.761	decaf::net::ssl::SSLSocket Class Reference	3337
6.761.1	Detailed Description	3339
6.761.2	Constructor & Destructor Documentation	3339
6.761.2.1	SSLSocket	3339
6.761.2.2	SSLSocket	3339
6.761.2.3	SSLSocket	3339
6.761.2.4	SSLSocket	3340
6.761.2.5	SSLSocket	3340
6.761.2.6	~SSLSocket	3340
6.761.3	Member Function Documentation	3340
6.761.3.1	getEnabledCipherSuites	3340
6.761.3.2	getEnabledProtocols	3341
6.761.3.3	getNeedClientAuth	3341
6.761.3.4	getSSLParameters	3341
6.761.3.5	getSupportedCipherSuites	3341
6.761.3.6	getSupportedProtocols	3342
6.761.3.7	getUseClientMode	3342
6.761.3.8	getWantClientAuth	3342
6.761.3.9	setEnabledCipherSuites	3342
6.761.3.10	setEnabledProtocols	3343
6.761.3.11	setNeedClientAuth	3343
6.761.3.12	setSSLParameters	3343
6.761.3.13	setUseClientMode	3344
6.761.3.14	setWantClientAuth	3344
6.761.3.15	startHandshake	3344
6.762	decaf::net::ssl::SSLSocketFactory Class Reference	3345
6.762.1	Detailed Description	3345
6.762.2	Constructor & Destructor Documentation	3346
6.762.2.1	SSLSocketFactory	3346
6.762.2.2	~SSLSocketFactory	3346
6.762.3	Member Function Documentation	3346
6.762.3.1	createSocket	3346
6.762.3.2	getDefault	3346
6.762.3.3	getDefaultCipherSuites	3347
6.762.3.4	getSupportedCipherSuites	3347

6.763activemq::transport::tcp::SslTransport Class Reference	3347
6.763.1 Detailed Description	3348
6.763.2 Constructor & Destructor Documentation	3348
6.763.2.1 SslTransport	3348
6.763.2.2 ~SslTransport	3349
6.763.3 Member Function Documentation	3349
6.763.3.1 configureSocket	3349
6.763.3.2 createSocket	3349
6.764activemq::transport::tcp::SslTransportFactory Class Reference	3349
6.764.1 Constructor & Destructor Documentation	3350
6.764.1.1 ~SslTransportFactory	3350
6.764.2 Member Function Documentation	3350
6.764.2.1 doCreateComposite	3350
6.765activemq::commands::BrokerError::StackTraceElement Struct Reference	3350
6.765.1 Field Documentation	3351
6.765.1.1 ClassName	3351
6.765.1.2 FileName	3351
6.765.1.3 LineNumber	3351
6.765.1.4 MethodName	3351
6.766decaf::internal::io::StandardErrorOutputStream Class Reference	3351
6.766.1 Detailed Description	3351
6.766.2 Constructor & Destructor Documentation	3352
6.766.2.1 StandardErrorOutputStream	3352
6.766.2.2 ~StandardErrorOutputStream	3352
6.766.3 Member Function Documentation	3352
6.766.3.1 close	3352
6.766.3.2 doWriteArrayBounded	3352
6.766.3.3 doWriteByte	3352
6.766.3.4 flush	3352
6.767decaf::internal::io::StandardInputStream Class Reference	3352
6.767.1 Constructor & Destructor Documentation	3353
6.767.1.1 StandardInputStream	3353
6.767.1.2 ~StandardInputStream	3353
6.767.2 Member Function Documentation	3353
6.767.2.1 available	3353
6.767.2.2 doReadByte	3353

6.768	decaf::internal::io::StandardOutputStream Class Reference	3354
6.768.1	Constructor & Destructor Documentation	3354
6.768.1.1	StandardOutputStream	3354
6.768.1.2	~StandardOutputStream	3354
6.768.2	Member Function Documentation	3354
6.768.2.1	close	3354
6.768.2.2	doWriteArrayBounded	3355
6.768.2.3	doWriteByte	3355
6.768.2.4	flush	3355
6.769	cms::Startable Class Reference	3355
6.769.1	Detailed Description	3355
6.769.2	Constructor & Destructor Documentation	3356
6.769.2.1	~Startable	3356
6.769.3	Member Function Documentation	3356
6.769.3.1	start	3356
6.770	decaf::lang::STATIC_CAST_TOKEN Struct Reference	3356
6.771	activemq::core::ActiveMQConstants::StaticInitializer Class Reference	3356
6.771.1	Constructor & Destructor Documentation	3357
6.771.1.1	StaticInitializer	3357
6.771.1.2	~StaticInitializer	3357
6.771.2	Field Documentation	3357
6.771.2.1	destOptionMap	3357
6.771.2.2	destOptions	3357
6.771.2.3	uriParams	3357
6.771.2.4	uriParamsMap	3357
6.772	decaf::util::StlList< E > Class Template Reference	3357
6.772.1	Detailed Description	3362
6.772.2	Constructor & Destructor Documentation	3362
6.772.2.1	StlList	3362
6.772.2.2	StlList	3362
6.772.2.3	StlList	3362
6.772.2.4	~StlList	3363
6.772.3	Member Function Documentation	3363
6.772.3.1	add	3363
6.772.3.2	add	3363
6.772.3.3	addAll	3364

6.772.3.4 clear	3364
6.772.3.5 contains	3365
6.772.3.6 copy	3365
6.772.3.7 equals	3365
6.772.3.8 get	3365
6.772.3.9 indexOf	3366
6.772.3.10isEmpty	3366
6.772.3.11iterator	3366
6.772.3.12iterator	3366
6.772.3.13lastIndexOf	3366
6.772.3.14listIterator	3367
6.772.3.15listIterator	3367
6.772.3.16listIterator	3367
6.772.3.17listIterator	3367
6.772.3.18remove	3368
6.772.3.19remove	3368
6.772.3.20set	3369
6.772.3.21size	3369
6.773decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference	3369
6.773.1 Detailed Description	3373
6.773.2 Constructor & Destructor Documentation	3373
6.773.2.1 StlMap	3373
6.773.2.2 StlMap	3373
6.773.2.3 StlMap	3374
6.773.2.4 ~StlMap	3374
6.773.3 Member Function Documentation	3374
6.773.3.1 clear	3374
6.773.3.2 containsKey	3374
6.773.3.3 containsValue	3374
6.773.3.4 copy	3375
6.773.3.5 copy	3375
6.773.3.6 equals	3375
6.773.3.7 equals	3376
6.773.3.8 get	3376
6.773.3.9 get	3376
6.773.3.10isEmpty	3377

6.773.3.1	keySet	3377
6.773.3.12	lock	3377
6.773.3.13	notify	3377
6.773.3.14	notifyAll	3378
6.773.3.15	put	3378
6.773.3.16	putAll	3378
6.773.3.17	putAll	3379
6.773.3.18	remove	3379
6.773.3.19	size	3379
6.773.3.20	tryLock	3380
6.773.3.21	unlock	3380
6.773.3.22	values	3380
6.773.3.23	wait	3380
6.773.3.24	wait	3381
6.773.3.25	wait	3381
6.774	decaf::util::StlQueue< T > Class Template Reference	3382
6.774.1	Detailed Description	3384
6.774.2	Constructor & Destructor Documentation	3384
6.774.2.1	StlQueue	3384
6.774.2.2	~StlQueue	3384
6.774.3	Member Function Documentation	3384
6.774.3.1	back	3384
6.774.3.2	back	3385
6.774.3.3	clear	3385
6.774.3.4	empty	3385
6.774.3.5	enqueueFront	3385
6.774.3.6	front	3385
6.774.3.7	front	3385
6.774.3.8	getSafeValue	3386
6.774.3.9	iterator	3386
6.774.3.10	lock	3386
6.774.3.11	notify	3386
6.774.3.12	notifyAll	3387
6.774.3.13	pop	3387
6.774.3.14	push	3387
6.774.3.15	reverse	3387

6.774.3.16size	3387
6.774.3.17toArray	3388
6.774.3.18ryLock	3388
6.774.3.19unlock	3388
6.774.3.20wait	3388
6.774.3.21lwait	3389
6.774.3.22wait	3389
6.775decaf::util::StlSet< E > Class Template Reference	3390
6.775.1 Detailed Description	3392
6.775.2 Constructor & Destructor Documentation	3392
6.775.2.1 StlSet	3392
6.775.2.2 StlSet	3392
6.775.2.3 StlSet	3392
6.775.2.4 ~StlSet	3392
6.775.3 Member Function Documentation	3392
6.775.3.1 add	3392
6.775.3.2 clear	3393
6.775.3.3 contains	3394
6.775.3.4 copy	3394
6.775.3.5 equals	3394
6.775.3.6 isEmpty	3394
6.775.3.7 iterator	3394
6.775.3.8 iterator	3394
6.775.3.9 remove	3394
6.775.3.10size	3395
6.776activemq::wireformat::stomp::StompCommandConstants Class Reference	3395
6.776.1 Field Documentation	3397
6.776.1.1 ABORT	3397
6.776.1.2 ACK	3397
6.776.1.3 ACK_AUTO	3397
6.776.1.4 ACK_CLIENT	3397
6.776.1.5 ACK_INDIVIDUAL	3397
6.776.1.6 BEGIN	3397
6.776.1.7 BYTES	3397
6.776.1.8 COMMIT	3397
6.776.1.9 CONNECT	3397

6.776.1.10	CONNECTED	3397
6.776.1.11	DISCONNECT	3397
6.776.1.12	ERROR_CMD	3397
6.776.1.13	HEADER_ACK	3397
6.776.1.14	HEADER_CLIENT_ID	3397
6.776.1.15	HEADER_CONSUMERPRIORITY	3397
6.776.1.16	HEADER_CONTENTLENGTH	3397
6.776.1.17	HEADER_CORRELATIONID	3397
6.776.1.18	HEADER_DESTINATION	3397
6.776.1.19	HEADER_DISPATCH_ASYNC	3397
6.776.1.20	HEADER_EXCLUSIVE	3397
6.776.1.21	HEADER_EXPIRES	3397
6.776.1.22	HEADER_ID	3397
6.776.1.23	HEADER_JMSPRIORITY	3397
6.776.1.24	HEADER_LOGIN	3397
6.776.1.25	HEADER_MAXPENDINGMSGLIMIT	3397
6.776.1.26	HEADER_MESSAGE	3397
6.776.1.27	HEADER_MESSAGEID	3397
6.776.1.28	HEADER_NOLOCAL	3397
6.776.1.29	HEADER_OLDSUBSCRIPTIONNAME	3397
6.776.1.30	HEADER_PASSWORD	3397
6.776.1.31	HEADER_PERSISTENT	3397
6.776.1.32	HEADER_PREFETCHSIZE	3397
6.776.1.33	HEADER_RECEIPT_REQUIRED	3397
6.776.1.34	HEADER_RECEIPTID	3397
6.776.1.35	HEADER_REDELIVERED	3397
6.776.1.36	HEADER_REDELIVERYCOUNT	3397
6.776.1.37	HEADER_REPLYTO	3397
6.776.1.38	HEADER_REQUESTID	3397
6.776.1.39	HEADER_RESPONSEID	3397
6.776.1.40	HEADER_RETROACTIVE	3397
6.776.1.41	HEADER_SELECTOR	3397
6.776.1.42	HEADER_SESSIONID	3397
6.776.1.43	HEADER_SUBSCRIPTION	3397
6.776.1.44	HEADER_SUBSCRIPTIONNAME	3397
6.776.1.45	HEADER_TIMESTAMP	3397

6.776.1.46	HEADER_TRANSACTIONID	3397
6.776.1.47	HEADER_TRANSFORMATION	3397
6.776.1.48	HEADER_TRANSFORMATION_ERROR	3397
6.776.1.49	HEADER_TYPE	3397
6.776.1.50	MESSAGE	3397
6.776.1.51	QUEUE_PREFIX	3397
6.776.1.52	RECEIPT	3397
6.776.1.53	SEND	3397
6.776.1.54	SUBSCRIBE	3397
6.776.1.55	TEMPQUEUE_PREFIX	3397
6.776.1.56	TEMPTOPIC_PREFIX	3397
6.776.1.57	TEXT	3397
6.776.1.58	TOPIC_PREFIX	3397
6.776.1.59	UNSUBSCRIBE	3397
6.777	activemq::wireformat::stomp::StompFrame Class Reference	3398
6.777.1	Detailed Description	3399
6.777.2	Constructor & Destructor Documentation	3399
6.777.2.1	StompFrame	3399
6.777.2.2	~StompFrame	3399
6.777.3	Member Function Documentation	3399
6.777.3.1	clone	3399
6.777.3.2	copy	3399
6.777.3.3	fromStream	3400
6.777.3.4	getBody	3400
6.777.3.5	getBody	3400
6.777.3.6	getBodyLength	3400
6.777.3.7	getCommand	3400
6.777.3.8	getProperties	3401
6.777.3.9	getProperties	3401
6.777.3.10	getProperty	3401
6.777.3.11	hasProperty	3401
6.777.3.12	removeProperty	3401
6.777.3.13	setBody	3402
6.777.3.14	setCommand	3402
6.777.3.15	setProperty	3402
6.777.3.16	oStream	3402

6.778	activemq::wireformat::stomp::StompHelper Class Reference	3402
6.778.1	Detailed Description	3403
6.778.2	Constructor & Destructor Documentation	3404
6.778.2.1	StompHelper	3404
6.778.2.2	~StompHelper	3404
6.778.3	Member Function Documentation	3404
6.778.3.1	convertConsumerId	3404
6.778.3.2	convertConsumerId	3404
6.778.3.3	convertDestination	3404
6.778.3.4	convertDestination	3405
6.778.3.5	convertMessageId	3405
6.778.3.6	convertMessageId	3405
6.778.3.7	convertProducerId	3405
6.778.3.8	convertProducerId	3406
6.778.3.9	convertProperties	3406
6.778.3.10	convertProperties	3406
6.778.3.11	convertTransactionId	3406
6.778.3.12	convertTransactionId	3407
6.779	activemq::wireformat::stomp::StompWireFormat Class Reference	3407
6.779.1	Constructor & Destructor Documentation	3408
6.779.1.1	StompWireFormat	3408
6.779.1.2	~StompWireFormat	3408
6.779.2	Member Function Documentation	3408
6.779.2.1	createNegotiator	3408
6.779.2.2	getVersion	3408
6.779.2.3	hasNegotiator	3409
6.779.2.4	inReceive	3409
6.779.2.5	marshal	3409
6.779.2.6	setVersion	3409
6.779.2.7	unmarshal	3410
6.780	activemq::wireformat::stomp::StompWireFormatFactory Class Reference	3410
6.780.1	Detailed Description	3410
6.780.2	Constructor & Destructor Documentation	3411
6.780.2.1	StompWireFormatFactory	3411
6.780.2.2	~StompWireFormatFactory	3411
6.780.3	Member Function Documentation	3411

6.780.3.1 createWireFormat	3411
6.781cms::Stoppable Class Reference	3411
6.781.1 Detailed Description	3411
6.781.2 Constructor & Destructor Documentation	3412
6.781.2.1 ~Stoppable	3412
6.781.3 Member Function Documentation	3412
6.781.3.1 stop	3412
6.782decaf::util::logging::StreamHandler Class Reference	3412
6.782.1 Detailed Description	3413
6.782.2 Constructor & Destructor Documentation	3413
6.782.2.1 StreamHandler	3413
6.782.2.2 StreamHandler	3413
6.782.2.3 ~StreamHandler	3414
6.782.3 Member Function Documentation	3414
6.782.3.1 close	3414
6.782.3.2 close	3414
6.782.3.3 flush	3414
6.782.3.4 isLoggable	3414
6.782.3.5 publish	3415
6.782.3.6 setOutputStream	3415
6.783cms::StreamMessage Class Reference	3415
6.783.1 Detailed Description	3418
6.783.2 Constructor & Destructor Documentation	3418
6.783.2.1 ~StreamMessage	3418
6.783.3 Member Function Documentation	3418
6.783.3.1 readBoolean	3418
6.783.3.2 readByte	3419
6.783.3.3 readBytes	3419
6.783.3.4 readBytes	3420
6.783.3.5 readChar	3420
6.783.3.6 readDouble	3421
6.783.3.7 readFloat	3421
6.783.3.8 readInt	3421
6.783.3.9 readLong	3422
6.783.3.10readShort	3422
6.783.3.11readString	3422

6.783.3.12	readUnsignedShort	3423
6.783.3.13	writeBoolean	3423
6.783.3.14	writeByte	3424
6.783.3.15	writeBytes	3424
6.783.3.16	writeBytes	3424
6.783.3.17	writeChar	3425
6.783.3.18	writeDouble	3425
6.783.3.19	writeFloat	3425
6.783.3.20	writeInt	3426
6.783.3.21	writeLong	3426
6.783.3.22	writeShort	3426
6.783.3.23	writeString	3427
6.783.3.24	writeUnsignedShort	3427
6.784	decaf::lang::String Class Reference	3427
6.784.1	Detailed Description	3428
6.784.2	Constructor & Destructor Documentation	3429
6.784.2.1	String	3429
6.784.2.2	String	3429
6.784.2.3	~String	3429
6.784.3	Member Function Documentation	3429
6.784.3.1	charAt	3429
6.784.3.2	isEmpty	3429
6.784.3.3	length	3429
6.784.3.4	subSequence	3430
6.784.3.5	toString	3430
6.785	decaf::util::StringTokenizer Class Reference	3430
6.785.1	Constructor & Destructor Documentation	3431
6.785.1.1	StringTokenizer	3431
6.785.1.2	~StringTokenizer	3431
6.785.2	Member Function Documentation	3431
6.785.2.1	countTokens	3431
6.785.2.2	hasMoreTokens	3432
6.785.2.3	nextToken	3432
6.785.2.4	nextToken	3432
6.785.2.5	reset	3433
6.785.2.6	toArray	3433

6.786	activemq::commands::SubscriptionInfo Class Reference	3433
6.786.1	Constructor & Destructor Documentation	3435
6.786.1.1	SubscriptionInfo	3435
6.786.1.2	~SubscriptionInfo	3435
6.786.2	Member Function Documentation	3435
6.786.2.1	cloneDataStructure	3435
6.786.2.2	copyDataStructure	3435
6.786.2.3	equals	3435
6.786.2.4	getClientId	3436
6.786.2.5	getClientId	3436
6.786.2.6	getDataStructureType	3436
6.786.2.7	getDestination	3437
6.786.2.8	getDestination	3437
6.786.2.9	getSelector	3437
6.786.2.10	getSelector	3437
6.786.2.11	getSubscriptionName	3437
6.786.2.12	getSubscriptionName	3437
6.786.2.13	getSubscribedDestination	3437
6.786.2.14	getSubscribedDestination	3437
6.786.2.15	setClientId	3437
6.786.2.16	setDestination	3437
6.786.2.17	setSelector	3437
6.786.2.18	setSubscriptionName	3437
6.786.2.19	setSubscribedDestination	3437
6.786.2.20	toString	3437
6.786.3	Field Documentation	3438
6.786.3.1	clientId	3438
6.786.3.2	destination	3438
6.786.3.3	ID_SUBSCRIPTIONINFO	3438
6.786.3.4	selector	3438
6.786.3.5	subscriptionName	3438
6.786.3.6	subscribedDestination	3438
6.787	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller Class Reference	3438
6.787.1	Detailed Description	3439
6.787.2	Constructor & Destructor Documentation	3439
6.787.2.1	SubscriptionInfoMarshaller	3439

6.787.2.2 ~SubscriptionInfoMarshaller	3439
6.787.3 Member Function Documentation	3439
6.787.3.1 createObject	3439
6.787.3.2 getDataStructureType	3440
6.787.3.3 looseMarshal	3440
6.787.3.4 looseUnmarshal	3440
6.787.3.5 tightMarshal1	3441
6.787.3.6 tightMarshal2	3441
6.787.3.7 tightUnmarshal	3442
6.788activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller Class	
Reference	3442
6.788.1 Detailed Description	3443
6.788.2 Constructor & Destructor Documentation	3443
6.788.2.1 SubscriptionInfoMarshaller	3443
6.788.2.2 ~SubscriptionInfoMarshaller	3443
6.788.3 Member Function Documentation	3443
6.788.3.1 createObject	3443
6.788.3.2 getDataStructureType	3444
6.788.3.3 looseMarshal	3444
6.788.3.4 looseUnmarshal	3444
6.788.3.5 tightMarshal1	3445
6.788.3.6 tightMarshal2	3445
6.788.3.7 tightUnmarshal	3445
6.789activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller Class	
Reference	3446
6.789.1 Detailed Description	3447
6.789.2 Constructor & Destructor Documentation	3447
6.789.2.1 SubscriptionInfoMarshaller	3447
6.789.2.2 ~SubscriptionInfoMarshaller	3447
6.789.3 Member Function Documentation	3447
6.789.3.1 createObject	3447
6.789.3.2 getDataStructureType	3447
6.789.3.3 looseMarshal	3448
6.789.3.4 looseUnmarshal	3448
6.789.3.5 tightMarshal1	3448
6.789.3.6 tightMarshal2	3449
6.789.3.7 tightUnmarshal	3449

6.790	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	Class	
	Reference		3450
6.790.1	Detailed Description		3451
6.790.2	Constructor & Destructor Documentation		3451
	6.790.2.1 SubscriptionInfoMarshaller		3451
	6.790.2.2 ~SubscriptionInfoMarshaller		3451
6.790.3	Member Function Documentation		3451
	6.790.3.1 createObject		3451
	6.790.3.2 getDataStructureType		3451
	6.790.3.3 looseMarshal		3451
	6.790.3.4 looseUnmarshal		3452
	6.790.3.5 tightMarshal1		3452
	6.790.3.6 tightMarshal2		3453
	6.790.3.7 tightUnmarshal		3453
6.791	activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller	Class	
	Reference		3453
6.791.1	Detailed Description		3454
6.791.2	Constructor & Destructor Documentation		3455
	6.791.2.1 SubscriptionInfoMarshaller		3455
	6.791.2.2 ~SubscriptionInfoMarshaller		3455
6.791.3	Member Function Documentation		3455
	6.791.3.1 createObject		3455
	6.791.3.2 getDataStructureType		3455
	6.791.3.3 looseMarshal		3455
	6.791.3.4 looseUnmarshal		3456
	6.791.3.5 tightMarshal1		3456
	6.791.3.6 tightMarshal2		3456
	6.791.3.7 tightUnmarshal		3457
6.792	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	Class	
	Reference		3457
6.792.1	Detailed Description		3458
6.792.2	Constructor & Destructor Documentation		3459
	6.792.2.1 SubscriptionInfoMarshaller		3459
	6.792.2.2 ~SubscriptionInfoMarshaller		3459
6.792.3	Member Function Documentation		3459
	6.792.3.1 createObject		3459
	6.792.3.2 getDataStructureType		3459

6.792.3.3 looseMarshal	3459
6.792.3.4 looseUnmarshal	3460
6.792.3.5 tightMarshal1	3460
6.792.3.6 tightMarshal2	3460
6.792.3.7 tightUnmarshal	3461
6.793decaf::util::concurrent::Synchronizable Class Reference	3461
6.793.1 Detailed Description	3462
6.793.2 Constructor & Destructor Documentation	3463
6.793.2.1 ~Synchronizable	3463
6.793.3 Member Function Documentation	3463
6.793.3.1 lock	3463
6.793.3.2 notify	3464
6.793.3.3 notifyAll	3465
6.793.3.4 tryLock	3466
6.793.3.5 unlock	3467
6.793.3.6 wait	3468
6.793.3.7 wait	3470
6.793.3.8 wait	3471
6.794decaf::internal::util::concurrent::SynchronizableImpl Class Reference	3473
6.794.1 Detailed Description	3474
6.794.2 Constructor & Destructor Documentation	3474
6.794.2.1 SynchronizableImpl	3474
6.794.2.2 ~SynchronizableImpl	3474
6.794.3 Member Function Documentation	3474
6.794.3.1 lock	3474
6.794.3.2 notify	3474
6.794.3.3 notifyAll	3474
6.794.3.4 tryLock	3475
6.794.3.5 unlock	3475
6.794.3.6 wait	3475
6.794.3.7 wait	3476
6.794.3.8 wait	3476
6.795activemq::core::Synchronization Class Reference	3477
6.795.1 Detailed Description	3477
6.795.2 Constructor & Destructor Documentation	3477
6.795.2.1 ~Synchronization	3477

6.795.3 Member Function Documentation	3477
6.795.3.1 afterCommit	3477
6.795.3.2 afterRollback	3477
6.795.3.3 beforeEnd	3477
6.796decaf::util::concurrent::SynchronousQueue< E > Class Template Reference	3477
6.796.1 Detailed Description	3479
6.796.2 Constructor & Destructor Documentation	3480
6.796.2.1 SynchronousQueue	3480
6.796.2.2 ~SynchronousQueue	3480
6.796.3 Member Function Documentation	3480
6.796.3.1 clear	3480
6.796.3.2 contains	3480
6.796.3.3 containsAll	3480
6.796.3.4 drainTo	3481
6.796.3.5 drainTo	3481
6.796.3.6 equals	3482
6.796.3.7 isEmpty	3482
6.796.3.8 iterator	3483
6.796.3.9 iterator	3483
6.796.3.10offer	3483
6.796.3.11offer	3483
6.796.3.12peek	3484
6.796.3.13poll	3484
6.796.3.14poll	3484
6.796.3.15put	3485
6.796.3.16remainingCapacity	3485
6.796.3.17remove	3486
6.796.3.18removeAll	3486
6.796.3.19retainAll	3486
6.796.3.20size	3486
6.796.3.21take	3486
6.796.3.22toArray	3486
6.797decaf::lang::System Class Reference	3487
6.797.1 Detailed Description	3489
6.797.2 Constructor & Destructor Documentation	3489
6.797.2.1 System	3489

6.797.2.2	~System	3489
6.797.3	Member Function Documentation	3489
6.797.3.1	arraycopy	3489
6.797.3.2	arraycopy	3489
6.797.3.3	arraycopy	3490
6.797.3.4	arraycopy	3490
6.797.3.5	availableProcessors	3491
6.797.3.6	clearProperty	3491
6.797.3.7	currentTimeMillis	3491
6.797.3.8	getenv	3491
6.797.3.9	getenv	3492
6.797.3.10	getProperties	3492
6.797.3.11	getProperty	3492
6.797.3.12	getProperty	3493
6.797.3.13	nanoTime	3493
6.797.3.14	setenv	3493
6.797.3.15	setProperty	3494
6.797.3.16	unsetenv	3494
6.797.4	Friends And Related Function Documentation	3494
6.797.4.1	decaf::lang::Runtime	3494
6.798	activemq::threads::Task Class Reference	3494
6.798.1	Detailed Description	3495
6.798.2	Constructor & Destructor Documentation	3495
6.798.2.1	~Task	3495
6.798.3	Member Function Documentation	3495
6.798.3.1	iterate	3495
6.799	decaf::util::concurrent::TaskListener Class Reference	3495
6.799.1	Constructor & Destructor Documentation	3496
6.799.1.1	~TaskListener	3496
6.799.2	Member Function Documentation	3496
6.799.2.1	onTaskComplete	3496
6.799.2.2	onTaskException	3496
6.800	activemq::threads::TaskRunner Class Reference	3496
6.800.1	Constructor & Destructor Documentation	3497
6.800.1.1	~TaskRunner	3497
6.800.2	Member Function Documentation	3497

6.800.2.1 shutdown	3497
6.800.2.2 shutdown	3497
6.800.2.3 wakeup	3497
6.801decaf::internal::net::tcp::TcpSocket Class Reference	3497
6.801.1 Detailed Description	3500
6.801.2 Constructor & Destructor Documentation	3500
6.801.2.1 TcpSocket	3500
6.801.2.2 ~TcpSocket	3501
6.801.3 Member Function Documentation	3501
6.801.3.1 accept	3501
6.801.3.2 available	3501
6.801.3.3 bind	3501
6.801.3.4 checkResult	3501
6.801.3.5 close	3501
6.801.3.6 connect	3502
6.801.3.7 create	3502
6.801.3.8 getInputStream	3502
6.801.3.9 getLocalAddress	3503
6.801.3.10getOption	3503
6.801.3.11getOutputStream	3503
6.801.3.12getSocketHandle	3503
6.801.3.13sClosed	3504
6.801.3.14sConnected	3504
6.801.3.15listen	3504
6.801.3.16read	3504
6.801.3.17setOption	3505
6.801.3.18shutdownInput	3505
6.801.3.19shutdownOutput	3505
6.801.3.20write	3505
6.802decaf::internal::net::tcp::TcpSocketInputStream Class Reference	3506
6.802.1 Detailed Description	3507
6.802.2 Constructor & Destructor Documentation	3507
6.802.2.1 TcpSocketInputStream	3507
6.802.2.2 ~TcpSocketInputStream	3507
6.802.3 Member Function Documentation	3507
6.802.3.1 available	3507

6.802.3.2 close	3508
6.802.3.3 doReadArrayBounded	3508
6.802.3.4 doReadByte	3508
6.802.3.5 skip	3508
6.803decaf::internal::net::tcp::TcpSocketOutputStream Class Reference	3509
6.803.1 Detailed Description	3509
6.803.2 Constructor & Destructor Documentation	3509
6.803.2.1 TcpSocketOutputStream	3509
6.803.2.2 ~TcpSocketOutputStream	3510
6.803.3 Member Function Documentation	3510
6.803.3.1 close	3510
6.803.3.2 doWriteArrayBounded	3510
6.803.3.3 doWriteByte	3510
6.804activemq::transport::tcp::TcpTransport Class Reference	3510
6.804.1 Detailed Description	3511
6.804.2 Constructor & Destructor Documentation	3511
6.804.2.1 TcpTransport	3511
6.804.2.2 ~TcpTransport	3512
6.804.3 Member Function Documentation	3512
6.804.3.1 close	3512
6.804.3.2 configureSocket	3512
6.804.3.3 connect	3512
6.804.3.4 createSocket	3513
6.804.3.5 isClosed	3513
6.804.3.6 isConnected	3513
6.804.3.7 isFaultTolerant	3513
6.805activemq::transport::tcp::TcpTransportFactory Class Reference	3514
6.805.1 Detailed Description	3514
6.805.2 Constructor & Destructor Documentation	3515
6.805.2.1 ~TcpTransportFactory	3515
6.805.3 Member Function Documentation	3515
6.805.3.1 create	3515
6.805.3.2 createComposite	3515
6.805.3.3 doCreateComposite	3515
6.806cms::TemporaryQueue Class Reference	3516
6.806.1 Detailed Description	3516

6.806.2 Constructor & Destructor Documentation	3517
6.806.2.1 ~TemporaryQueue	3517
6.806.3 Member Function Documentation	3517
6.806.3.1 destroy	3517
6.806.3.2 getQueueName	3517
6.807cms::TemporaryTopic Class Reference	3517
6.807.1 Detailed Description	3518
6.807.2 Constructor & Destructor Documentation	3518
6.807.2.1 ~TemporaryTopic	3518
6.807.3 Member Function Documentation	3518
6.807.3.1 destroy	3518
6.807.3.2 getTopicName	3518
6.808cms::TextMessage Class Reference	3519
6.808.1 Detailed Description	3519
6.808.2 Constructor & Destructor Documentation	3519
6.808.2.1 ~TextMessage	3519
6.808.3 Member Function Documentation	3519
6.808.3.1 getText	3519
6.808.3.2 setText	3520
6.808.3.3 setText	3520
6.809decaf::lang::Thread Class Reference	3520
6.809.1 Detailed Description	3523
6.809.2 Member Enumeration Documentation	3523
6.809.2.1 State	3523
6.809.3 Constructor & Destructor Documentation	3524
6.809.3.1 Thread	3524
6.809.3.2 Thread	3524
6.809.3.3 Thread	3524
6.809.3.4 Thread	3524
6.809.3.5 ~Thread	3525
6.809.4 Member Function Documentation	3525
6.809.4.1 currentThread	3525
6.809.4.2 getId	3525
6.809.4.3 getName	3525
6.809.4.4 getPriority	3525
6.809.4.5 getState	3525

6.809.4.6	getUncaughtExceptionHandler	3526
6.809.4.7	isAlive	3526
6.809.4.8	join	3526
6.809.4.9	join	3526
6.809.4.10	join	3527
6.809.4.11	run	3527
6.809.4.12	setName	3527
6.809.4.13	setPriority	3527
6.809.4.14	setUncaughtExceptionHandler	3527
6.809.4.15	sleep	3528
6.809.4.16	sleep	3528
6.809.4.17	start	3528
6.809.4.18	toString	3529
6.809.4.19	yield	3529
6.809.5	Friends And Related Function Documentation	3529
6.809.5.1	decaf::lang::Runtime	3529
6.809.5.2	decaf::util::concurrent::locks::LockSupport	3529
6.809.6	Field Documentation	3529
6.809.6.1	MAX_PRIORITY	3529
6.809.6.2	MIN_PRIORITY	3529
6.809.6.3	NORM_PRIORITY	3529
6.810	decaf::util::concurrent::ThreadFactory Class Reference	3529
6.810.1	Detailed Description	3530
6.810.2	Constructor & Destructor Documentation	3530
6.810.2.1	~ThreadFactory	3530
6.810.3	Member Function Documentation	3530
6.810.3.1	newThread	3530
6.811	decaf::lang::ThreadGroup Class Reference	3531
6.811.1	Detailed Description	3531
6.811.2	Constructor & Destructor Documentation	3531
6.811.2.1	ThreadGroup	3531
6.811.2.2	~ThreadGroup	3531
6.812	decaf::util::concurrent::ThreadPool Class Reference	3531
6.812.1	Detailed Description	3533
6.812.2	Member Typedef Documentation	3533
6.812.2.1	Task	3533

6.812.3 Constructor & Destructor Documentation	3533
6.812.3.1 ThreadPool	3533
6.812.3.2 ~ThreadPool	3533
6.812.4 Member Function Documentation	3533
6.812.4.1 deQueueTask	3533
6.812.4.2 getBacklog	3533
6.812.4.3 getBlockSize	3534
6.812.4.4 getFreeThreadCount	3534
6.812.4.5 getInstance	3534
6.812.4.6 getMaxThreads	3534
6.812.4.7 getPoolSize	3534
6.812.4.8 onTaskCompleted	3535
6.812.4.9 onTaskException	3535
6.812.4.10 onTaskStarted	3535
6.812.4.11 queueTask	3535
6.812.4.12 reserve	3536
6.812.4.13 setBlockSize	3536
6.812.4.14 setMaxThreads	3536
6.812.5 Field Documentation	3536
6.812.5.1 DEFAULT_MAX_BLOCK_SIZE	3536
6.812.5.2 DEFAULT_MAX_POOL_SIZE	3536
6.813 decaf::lang::Throwable Class Reference	3536
6.813.1 Detailed Description	3537
6.813.2 Constructor & Destructor Documentation	3538
6.813.2.1 Throwable	3538
6.813.2.2 ~Throwable	3538
6.813.3 Member Function Documentation	3538
6.813.3.1 clone	3538
6.813.3.2 getCause	3539
6.813.3.3 getMessage	3539
6.813.3.4 getStackTrace	3539
6.813.3.5 getStackTraceString	3539
6.813.3.6 initCause	3540
6.813.3.7 printStackTrace	3540
6.813.3.8 printStackTrace	3540
6.813.3.9 setMark	3540

6.814decaf::util::concurrent::TimeoutException Class Reference	3541
6.814.1 Constructor & Destructor Documentation	3541
6.814.1.1 TimeoutException	3541
6.814.1.2 TimeoutException	3541
6.814.1.3 TimeoutException	3542
6.814.1.4 TimeoutException	3542
6.814.1.5 TimeoutException	3542
6.814.1.6 TimeoutException	3542
6.814.1.7 ~TimeoutException	3543
6.814.2 Member Function Documentation	3543
6.814.2.1 clone	3543
6.815decaf::util::Timer Class Reference	3543
6.815.1 Detailed Description	3545
6.815.2 Constructor & Destructor Documentation	3545
6.815.2.1 Timer	3545
6.815.2.2 ~Timer	3545
6.815.3 Member Function Documentation	3545
6.815.3.1 cancel	3545
6.815.3.2 purge	3545
6.815.3.3 schedule	3546
6.815.3.4 schedule	3546
6.815.3.5 schedule	3547
6.815.3.6 schedule	3548
6.815.3.7 schedule	3548
6.815.3.8 schedule	3549
6.815.3.9 schedule	3550
6.815.3.10 schedule	3550
6.815.3.11 scheduleAtFixedRate	3551
6.815.3.12 scheduleAtFixedRate	3552
6.815.3.13 scheduleAtFixedRate	3552
6.815.3.14 scheduleAtFixedRate	3553
6.816decaf::util::TimerTask Class Reference	3554
6.816.1 Detailed Description	3555
6.816.2 Constructor & Destructor Documentation	3555
6.816.2.1 TimerTask	3555
6.816.2.2 ~TimerTask	3555

6.816.3 Member Function Documentation	3555
6.816.3.1 cancel	3555
6.816.3.2 getWhen	3555
6.816.3.3 isScheduled	3555
6.816.3.4 scheduledExecutionTime	3555
6.816.3.5 setScheduledTime	3556
6.816.4 Friends And Related Function Documentation	3556
6.816.4.1 decaf::internal::util::TimerTaskHeap	3556
6.816.4.2 Timer	3556
6.816.4.3 TimerImpl	3556
6.817 decaf::internal::util::TimerTaskHeap Class Reference	3556
6.817.1 Detailed Description	3557
6.817.2 Constructor & Destructor Documentation	3557
6.817.2.1 TimerTaskHeap	3557
6.817.2.2 ~TimerTaskHeap	3557
6.817.3 Member Function Documentation	3557
6.817.3.1 adjustMinimum	3557
6.817.3.2 deleteIfCancelled	3557
6.817.3.3 find	3558
6.817.3.4 insert	3558
6.817.3.5 isEmpty	3558
6.817.3.6 peek	3558
6.817.3.7 remove	3558
6.817.3.8 reset	3558
6.817.3.9 size	3559
6.818 decaf::util::concurrent::TimeUnit Class Reference	3559
6.818.1 Detailed Description	3561
6.818.2 Constructor & Destructor Documentation	3561
6.818.2.1 TimeUnit	3561
6.818.2.2 ~TimeUnit	3561
6.818.3 Member Function Documentation	3561
6.818.3.1 compareTo	3561
6.818.3.2 convert	3562
6.818.3.3 equals	3562
6.818.3.4 operator<	3563
6.818.3.5 operator==	3563

6.818.3.6 sleep	3563
6.818.3.7 timedJoin	3563
6.818.3.8 timedWait	3564
6.818.3.9 toDays	3564
6.818.3.10 toHours	3565
6.818.3.11 toMicros	3565
6.818.3.12 toMillis	3565
6.818.3.13 toMinutes	3566
6.818.3.14 toNanos	3566
6.818.3.15 toSeconds	3566
6.818.3.16 toString	3567
6.818.3.17 valueOf	3567
6.818.4 Field Documentation	3567
6.818.4.1 DAYS	3567
6.818.4.2 HOURS	3567
6.818.4.3 MICROSECONDS	3567
6.818.4.4 MILLISECONDS	3567
6.818.4.5 MINUTES	3567
6.818.4.6 NANOSECONDS	3567
6.818.4.7 SECONDS	3568
6.818.4.8 values	3568
6.819 cms::Topic Class Reference	3568
6.819.1 Detailed Description	3568
6.819.2 Constructor & Destructor Documentation	3568
6.819.2.1 ~Topic	3568
6.819.3 Member Function Documentation	3568
6.819.3.1 getTopicName	3568
6.820 activemq::state::Tracked Class Reference	3569
6.820.1 Constructor & Destructor Documentation	3569
6.820.1.1 Tracked	3569
6.820.1.2 Tracked	3569
6.820.1.3 ~Tracked	3569
6.820.2 Member Function Documentation	3569
6.820.2.1 isWaitingForResponse	3569
6.820.2.2 onResponse	3569
6.821 activemq::commands::TransactionId Class Reference	3569

6.821.1 Member Typedef Documentation	3571
6.821.1.1 COMPARATOR	3571
6.821.2 Constructor & Destructor Documentation	3571
6.821.2.1 TransactionId	3571
6.821.2.2 TransactionId	3571
6.821.2.3 ~TransactionId	3571
6.821.3 Member Function Documentation	3571
6.821.3.1 cloneDataStructure	3571
6.821.3.2 compareTo	3571
6.821.3.3 copyDataStructure	3571
6.821.3.4 equals	3571
6.821.3.5 equals	3572
6.821.3.6 getDataStructureType	3572
6.821.3.7 operator<	3572
6.821.3.8 operator=	3572
6.821.3.9 operator==	3572
6.821.3.10 toString	3572
6.821.4 Field Documentation	3572
6.821.4.1 ID_TRANSACTIONID	3572
6.822activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller Class Reference	3572
6.822.1 Detailed Description	3573
6.822.2 Constructor & Destructor Documentation	3574
6.822.2.1 TransactionIdMarshaller	3574
6.822.2.2 ~TransactionIdMarshaller	3574
6.822.3 Member Function Documentation	3574
6.822.3.1 looseMarshal	3574
6.822.3.2 looseUnmarshal	3574
6.822.3.3 tightMarshal1	3575
6.822.3.4 tightMarshal2	3575
6.822.3.5 tightUnmarshal	3576
6.823activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller Class Reference	3576
6.823.1 Detailed Description	3577
6.823.2 Constructor & Destructor Documentation	3577
6.823.2.1 TransactionIdMarshaller	3577
6.823.2.2 ~TransactionIdMarshaller	3577

6.823.3 Member Function Documentation	3577
6.823.3.1 looseMarshal	3577
6.823.3.2 looseUnmarshal	3578
6.823.3.3 tightMarshal1	3578
6.823.3.4 tightMarshal2	3579
6.823.3.5 tightUnmarshal	3579
6.824activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller Class Reference	3580
6.824.1 Detailed Description	3581
6.824.2 Constructor & Destructor Documentation	3581
6.824.2.1 TransactionIdMarshaller	3581
6.824.2.2 ~TransactionIdMarshaller	3581
6.824.3 Member Function Documentation	3581
6.824.3.1 looseMarshal	3581
6.824.3.2 looseUnmarshal	3581
6.824.3.3 tightMarshal1	3582
6.824.3.4 tightMarshal2	3582
6.824.3.5 tightUnmarshal	3583
6.825activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller Class Reference	3583
6.825.1 Detailed Description	3584
6.825.2 Constructor & Destructor Documentation	3585
6.825.2.1 TransactionIdMarshaller	3585
6.825.2.2 ~TransactionIdMarshaller	3585
6.825.3 Member Function Documentation	3585
6.825.3.1 looseMarshal	3585
6.825.3.2 looseUnmarshal	3585
6.825.3.3 tightMarshal1	3586
6.825.3.4 tightMarshal2	3586
6.825.3.5 tightUnmarshal	3587
6.826activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller Class Reference	3587
6.826.1 Detailed Description	3588
6.826.2 Constructor & Destructor Documentation	3588
6.826.2.1 TransactionIdMarshaller	3588
6.826.2.2 ~TransactionIdMarshaller	3588
6.826.3 Member Function Documentation	3588

6.826.3.1 looseMarshal	3588
6.826.3.2 looseUnmarshal	3589
6.826.3.3 tightMarshal1	3589
6.826.3.4 tightMarshal2	3590
6.826.3.5 tightUnmarshal	3590
6.827activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller Class Reference	3591
6.827.1 Detailed Description	3592
6.827.2 Constructor & Destructor Documentation	3592
6.827.2.1 TransactionIdMarshaller	3592
6.827.2.2 ~TransactionIdMarshaller	3592
6.827.3 Member Function Documentation	3592
6.827.3.1 looseMarshal	3592
6.827.3.2 looseUnmarshal	3592
6.827.3.3 tightMarshal1	3593
6.827.3.4 tightMarshal2	3593
6.827.3.5 tightUnmarshal	3594
6.828activemq::commands::TransactionInfo Class Reference	3594
6.828.1 Constructor & Destructor Documentation	3596
6.828.1.1 TransactionInfo	3596
6.828.1.2 ~TransactionInfo	3596
6.828.2 Member Function Documentation	3596
6.828.2.1 cloneDataStructure	3596
6.828.2.2 copyDataStructure	3596
6.828.2.3 equals	3596
6.828.2.4 getConnectionId	3597
6.828.2.5 getConnectionId	3597
6.828.2.6 getDataStructureType	3597
6.828.2.7 getTransactionId	3597
6.828.2.8 getTransactionId	3597
6.828.2.9 getType	3597
6.828.2.10sTransactionInfo	3597
6.828.2.11setConnectionId	3597
6.828.2.12setTransactionId	3597
6.828.2.13setType	3597
6.828.2.14toString	3597
6.828.2.15visit	3598

6.828.3 Field Documentation	3598
6.828.3.1 connectionId	3598
6.828.3.2 ID_TRANSACTIONINFO	3598
6.828.3.3 transactionId	3598
6.828.3.4 type	3598
6.829activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller Class Reference	3598
6.829.1 Detailed Description	3599
6.829.2 Constructor & Destructor Documentation	3600
6.829.2.1 TransactionInfoMarshaller	3600
6.829.2.2 ~TransactionInfoMarshaller	3600
6.829.3 Member Function Documentation	3600
6.829.3.1 createObject	3600
6.829.3.2 getDataStructureType	3600
6.829.3.3 looseMarshal	3600
6.829.3.4 looseUnmarshal	3601
6.829.3.5 tightMarshal1	3601
6.829.3.6 tightMarshal2	3601
6.829.3.7 tightUnmarshal	3602
6.830activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller Class Reference	3602
6.830.1 Detailed Description	3603
6.830.2 Constructor & Destructor Documentation	3604
6.830.2.1 TransactionInfoMarshaller	3604
6.830.2.2 ~TransactionInfoMarshaller	3604
6.830.3 Member Function Documentation	3604
6.830.3.1 createObject	3604
6.830.3.2 getDataStructureType	3604
6.830.3.3 looseMarshal	3604
6.830.3.4 looseUnmarshal	3605
6.830.3.5 tightMarshal1	3605
6.830.3.6 tightMarshal2	3605
6.830.3.7 tightUnmarshal	3606
6.831activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller Class Reference	3606
6.831.1 Detailed Description	3607
6.831.2 Constructor & Destructor Documentation	3608

6.831.2.1 TransactionInfoMarshaller	3608
6.831.2.2 ~TransactionInfoMarshaller	3608
6.831.3 Member Function Documentation	3608
6.831.3.1 createObject	3608
6.831.3.2 getDataStructureType	3608
6.831.3.3 looseMarshal	3608
6.831.3.4 looseUnmarshal	3609
6.831.3.5 tightMarshal1	3609
6.831.3.6 tightMarshal2	3609
6.831.3.7 tightUnmarshal	3610
6.832activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller Class Reference	3610
6.832.1 Detailed Description	3611
6.832.2 Constructor & Destructor Documentation	3612
6.832.2.1 TransactionInfoMarshaller	3612
6.832.2.2 ~TransactionInfoMarshaller	3612
6.832.3 Member Function Documentation	3612
6.832.3.1 createObject	3612
6.832.3.2 getDataStructureType	3612
6.832.3.3 looseMarshal	3612
6.832.3.4 looseUnmarshal	3613
6.832.3.5 tightMarshal1	3613
6.832.3.6 tightMarshal2	3613
6.832.3.7 tightUnmarshal	3614
6.833activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller Class Reference	3614
6.833.1 Detailed Description	3615
6.833.2 Constructor & Destructor Documentation	3616
6.833.2.1 TransactionInfoMarshaller	3616
6.833.2.2 ~TransactionInfoMarshaller	3616
6.833.3 Member Function Documentation	3616
6.833.3.1 createObject	3616
6.833.3.2 getDataStructureType	3616
6.833.3.3 looseMarshal	3616
6.833.3.4 looseUnmarshal	3617
6.833.3.5 tightMarshal1	3617
6.833.3.6 tightMarshal2	3617

6.833.3.7 tightUnmarshal	3618
6.834activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller Class Reference	3618
6.834.1 Detailed Description	3619
6.834.2 Constructor & Destructor Documentation	3620
6.834.2.1 TransactionInfoMarshaller	3620
6.834.2.2 ~TransactionInfoMarshaller	3620
6.834.3 Member Function Documentation	3620
6.834.3.1 createObject	3620
6.834.3.2 getDataStructureType	3620
6.834.3.3 looseMarshal	3620
6.834.3.4 looseUnmarshal	3621
6.834.3.5 tightMarshal1	3621
6.834.3.6 tightMarshal2	3621
6.834.3.7 tightUnmarshal	3622
6.835activemq::state::TransactionState Class Reference	3622
6.835.1 Constructor & Destructor Documentation	3624
6.835.1.1 TransactionState	3624
6.835.1.2 ~TransactionState	3624
6.835.2 Member Function Documentation	3624
6.835.2.1 addCommand	3624
6.835.2.2 addProducerState	3624
6.835.2.3 checkShutdown	3624
6.835.2.4 getCommands	3624
6.835.2.5 getId	3624
6.835.2.6 getPreparedResult	3624
6.835.2.7 getProducerStates	3624
6.835.2.8 isPrepared	3624
6.835.2.9 setPrepared	3624
6.835.2.10setPreparedResult	3624
6.835.2.11shutdown	3624
6.835.2.12toString	3624
6.836decaf::internal::util::concurrent::Transferer< E > Class Template Reference	3625
6.836.1 Detailed Description	3625
6.837decaf::internal::util::concurrent::TransferQueue< E > Class Template Reference	3625
6.837.1 Detailed Description	3626
6.837.2 Constructor & Destructor Documentation	3626

6.837.2.1	TransferQueue	3626
6.837.2.2	~TransferQueue	3626
6.837.3	Member Function Documentation	3626
6.837.3.1	transfer	3626
6.837.3.2	transfer	3627
6.838	decaf::internal::util::concurrent::TransferStack< E > Class Template Reference	3627
6.838.1	Constructor & Destructor Documentation	3628
6.838.1.1	TransferStack	3628
6.838.1.2	~TransferStack	3628
6.838.2	Member Function Documentation	3628
6.838.2.1	transfer	3628
6.838.2.2	transfer	3628
6.839	activemq::transport::Transport Class Reference	3629
6.839.1	Detailed Description	3630
6.839.2	Constructor & Destructor Documentation	3630
6.839.2.1	~Transport	3630
6.839.3	Member Function Documentation	3630
6.839.3.1	getRemoteAddress	3630
6.839.3.2	getTransportListener	3631
6.839.3.3	isClosed	3631
6.839.3.4	isConnected	3631
6.839.3.5	isFaultTolerant	3631
6.839.3.6	narrow	3631
6.839.3.7	oneway	3632
6.839.3.8	reconnect	3632
6.839.3.9	request	3632
6.839.3.10	request	3633
6.839.3.11	setTransportListener	3633
6.839.3.12	setWireFormat	3633
6.839.3.13	start	3634
6.839.3.14	stop	3634
6.840	activemq::transport::TransportFactory Class Reference	3634
6.840.1	Detailed Description	3635
6.840.2	Constructor & Destructor Documentation	3635
6.840.2.1	~TransportFactory	3635
6.840.3	Member Function Documentation	3635

6.840.3.1 create	3635
6.840.3.2 createComposite	3635
6.841activemq::transport::TransportFilter Class Reference	3636
6.841.1 Detailed Description	3638
6.841.2 Constructor & Destructor Documentation	3638
6.841.2.1 TransportFilter	3638
6.841.2.2 ~TransportFilter	3638
6.841.3 Member Function Documentation	3638
6.841.3.1 close	3638
6.841.3.2 fire	3638
6.841.3.3 fire	3639
6.841.3.4 getRemoteAddress	3639
6.841.3.5 getTransportListener	3639
6.841.3.6 isClosed	3639
6.841.3.7 isConnected	3639
6.841.3.8 isFaultTolerant	3640
6.841.3.9 narrow	3640
6.841.3.10onCommand	3640
6.841.3.11oneway	3640
6.841.3.12onException	3641
6.841.3.13reconnect	3641
6.841.3.14request	3641
6.841.3.15request	3642
6.841.3.16setTransportListener	3642
6.841.3.17setWireFormat	3642
6.841.3.18start	3642
6.841.3.19stop	3643
6.841.3.20transportInterrupted	3643
6.841.3.21transportResumed	3643
6.841.4 Field Documentation	3643
6.841.4.1 listener	3643
6.841.4.2 next	3643
6.842activemq::transport::TransportListener Class Reference	3643
6.842.1 Detailed Description	3644
6.842.2 Constructor & Destructor Documentation	3644
6.842.2.1 ~TransportListener	3644

6.842.3 Member Function Documentation	3644
6.842.3.1 onCommand	3644
6.842.3.2 onException	3645
6.842.3.3 transportInterrupted	3645
6.842.3.4 transportResumed	3645
6.843activemq::transport::TransportRegistry Class Reference	3645
6.843.1 Detailed Description	3646
6.843.2 Constructor & Destructor Documentation	3646
6.843.2.1 ~TransportRegistry	3646
6.843.3 Member Function Documentation	3646
6.843.3.1 findFactory	3646
6.843.3.2 getInstance	3647
6.843.3.3 getTransportNames	3647
6.843.3.4 registerFactory	3647
6.843.3.5 unregisterFactory	3647
6.844tree_desc_s Struct Reference	3648
6.844.1 Field Documentation	3648
6.844.1.1 dyn_tree	3648
6.844.1.2 max_code	3648
6.844.1.3 stat_desc	3648
6.845decaf::lang::Thread::UncaughtExceptionHandler Class Reference	3648
6.845.1 Detailed Description	3649
6.845.2 Constructor & Destructor Documentation	3649
6.845.2.1 ~UncaughtExceptionHandler	3649
6.845.3 Member Function Documentation	3649
6.845.3.1 uncaughtException	3649
6.846decaf::net::UnknownHostException Class Reference	3649
6.846.1 Constructor & Destructor Documentation	3650
6.846.1.1 UnknownHostException	3650
6.846.1.2 UnknownHostException	3650
6.846.1.3 UnknownHostException	3650
6.846.1.4 UnknownHostException	3650
6.846.1.5 UnknownHostException	3651
6.846.1.6 UnknownHostException	3651
6.846.1.7 ~UnknownHostException	3651
6.846.2 Member Function Documentation	3651

6.846.2.1 clone	3651
6.847decaf::net::UnknownServiceException Class Reference	3652
6.847.1 Constructor & Destructor Documentation	3652
6.847.1.1 UnknownServiceException	3652
6.847.1.2 UnknownServiceException	3652
6.847.1.3 UnknownServiceException	3653
6.847.1.4 UnknownServiceException	3653
6.847.1.5 UnknownServiceException	3653
6.847.1.6 UnknownServiceException	3653
6.847.1.7 ~UnknownServiceException	3654
6.847.2 Member Function Documentation	3654
6.847.2.1 clone	3654
6.848decaf::io::UnsupportedEncodingException Class Reference	3654
6.848.1 Detailed Description	3655
6.848.2 Constructor & Destructor Documentation	3655
6.848.2.1 UnsupportedEncodingException	3655
6.848.2.2 UnsupportedEncodingException	3655
6.848.2.3 UnsupportedEncodingException	3655
6.848.2.4 UnsupportedEncodingException	3655
6.848.2.5 UnsupportedEncodingException	3656
6.848.2.6 UnsupportedEncodingException	3656
6.848.2.7 ~UnsupportedEncodingException	3656
6.848.3 Member Function Documentation	3656
6.848.3.1 clone	3656
6.849decaf::lang::exceptions::UnsupportedOperationException Class Reference	3657
6.849.1 Constructor & Destructor Documentation	3657
6.849.1.1 UnsupportedOperationException	3657
6.849.1.2 UnsupportedOperationException	3658
6.849.1.3 UnsupportedOperationException	3658
6.849.1.4 UnsupportedOperationException	3658
6.849.1.5 UnsupportedOperationException	3658
6.849.1.6 UnsupportedOperationException	3658
6.849.1.7 ~UnsupportedOperationException	3659
6.849.2 Member Function Documentation	3659
6.849.2.1 clone	3659
6.850cms::UnsupportedOperationException Class Reference	3659

6.850.1 Detailed Description	3660
6.850.2 Constructor & Destructor Documentation	3660
6.850.2.1 UnsupportedOperationException	3660
6.850.2.2 UnsupportedOperationException	3660
6.850.2.3 UnsupportedOperationException	3660
6.850.2.4 UnsupportedOperationException	3660
6.850.2.5 ~UnsupportedOperationException	3660
6.851decaf::net::URI Class Reference	3660
6.851.1 Detailed Description	3663
6.851.2 Constructor & Destructor Documentation	3663
6.851.2.1 URI	3663
6.851.2.2 URI	3663
6.851.2.3 URI	3663
6.851.2.4 URI	3663
6.851.2.5 URI	3664
6.851.2.6 URI	3664
6.851.2.7 URI	3664
6.851.2.8 ~URI	3665
6.851.3 Member Function Documentation	3665
6.851.3.1 compareTo	3665
6.851.3.2 create	3665
6.851.3.3 equals	3665
6.851.3.4 getAuthority	3665
6.851.3.5 getFragment	3665
6.851.3.6 getHost	3666
6.851.3.7 getPath	3666
6.851.3.8 getPort	3666
6.851.3.9 getQuery	3666
6.851.3.10getRawAuthority	3666
6.851.3.11getRawFragment	3666
6.851.3.12getRawPath	3667
6.851.3.13getRawQuery	3667
6.851.3.14getRawSchemeSpecificPart	3667
6.851.3.15getRawUserInfo	3667
6.851.3.16getScheme	3667
6.851.3.17getSchemeSpecificPart	3668

6.851.3.18	getUserInfo	3668
6.851.3.19	isAbsolute	3668
6.851.3.20	isOpaque	3668
6.851.3.21	normalize	3668
6.851.3.22	operator<	3669
6.851.3.23	operator==	3669
6.851.3.24	parseServerAuthority	3669
6.851.3.25	relativize	3670
6.851.3.26	resolve	3670
6.851.3.27	resolve	3670
6.851.3.28	toString	3671
6.851.3.29	toURL	3671
6.852	decaf::internal::net::URIEncoderDecoder Class Reference	3672
6.852.1	Constructor & Destructor Documentation	3673
6.852.1.1	URIEncoderDecoder	3673
6.852.1.2	~URIEncoderDecoder	3673
6.852.2	Member Function Documentation	3673
6.852.2.1	decode	3673
6.852.2.2	encodeOthers	3673
6.852.2.3	quoteIllegal	3673
6.852.2.4	validate	3674
6.852.2.5	validateSimple	3674
6.853	decaf::internal::net::URIHelper Class Reference	3674
6.853.1	Detailed Description	3676
6.853.2	Constructor & Destructor Documentation	3676
6.853.2.1	URIHelper	3676
6.853.2.2	URIHelper	3676
6.853.2.3	~URIHelper	3676
6.853.3	Member Function Documentation	3676
6.853.3.1	isValidDomainName	3676
6.853.3.2	isValidHexChar	3677
6.853.3.3	isValidHost	3677
6.853.3.4	isValidIP4Word	3677
6.853.3.5	isValidIP6Address	3678
6.853.3.6	isValidIPv4Address	3678
6.853.3.7	parseAuthority	3678

6.853.3.8	parseURI	3679
6.853.3.9	validateAuthority	3679
6.853.3.10	validateFragment	3679
6.853.3.11	validatePath	3680
6.853.3.12	validateQuery	3680
6.853.3.13	validateScheme	3680
6.853.3.14	validateSsp	3681
6.853.3.15	validateUserinfo	3681
6.854	activemq::transport::failover::URIPool Class Reference	3681
6.854.1	Constructor & Destructor Documentation	3682
6.854.1.1	URIPool	3682
6.854.1.2	URIPool	3682
6.854.1.3	~URIPool	3682
6.854.2	Member Function Documentation	3682
6.854.2.1	addURI	3682
6.854.2.2	addURIs	3683
6.854.2.3	getURI	3683
6.854.2.4	isRandomize	3683
6.854.2.5	removeURI	3683
6.854.2.6	setRandomize	3684
6.855	activemq::util::URISupport Class Reference	3684
6.855.1	Member Function Documentation	3685
6.855.1.1	createQueryString	3685
6.855.1.2	parseComposite	3685
6.855.1.3	parseQuery	3685
6.855.1.4	parseQuery	3686
6.855.1.5	parseURL	3686
6.856	decaf::net::URISyntaxException Class Reference	3686
6.856.1	Constructor & Destructor Documentation	3687
6.856.1.1	URISyntaxException	3687
6.856.1.2	URISyntaxException	3687
6.856.1.3	URISyntaxException	3688
6.856.1.4	URISyntaxException	3688
6.856.1.5	URISyntaxException	3688
6.856.1.6	URISyntaxException	3688
6.856.1.7	URISyntaxException	3689

6.856.1.8 URISyntaxException	3689
6.856.1.9 ~URISyntaxException	3689
6.856.2 Member Function Documentation	3689
6.856.2.1 clone	3689
6.856.2.2 getIndex	3690
6.856.2.3 getInput	3690
6.856.2.4 getReason	3690
6.857decaf::internal::net::URIType Class Reference	3690
6.857.1 Detailed Description	3692
6.857.2 Constructor & Destructor Documentation	3692
6.857.2.1 URIType	3692
6.857.2.2 URIType	3692
6.857.2.3 ~URIType	3692
6.857.3 Member Function Documentation	3692
6.857.3.1 getAuthority	3692
6.857.3.2 getFragment	3693
6.857.3.3 getHost	3693
6.857.3.4 getPath	3693
6.857.3.5 getPort	3693
6.857.3.6 getQuery	3693
6.857.3.7 getScheme	3693
6.857.3.8 getSchemeSpecificPart	3694
6.857.3.9 getSource	3694
6.857.3.10getUserInfo	3694
6.857.3.11isAbsolute	3694
6.857.3.12sOpaque	3694
6.857.3.13sServerAuthority	3694
6.857.3.14sValid	3695
6.857.3.15setAbsolute	3695
6.857.3.16setAuthority	3695
6.857.3.17setFragment	3695
6.857.3.18setHost	3695
6.857.3.19setOpaque	3695
6.857.3.20setPath	3696
6.857.3.21setPort	3696
6.857.3.22setQuery	3696

6.857.3.23	setScheme	3696
6.857.3.24	setSchemeSpecificPart	3696
6.857.3.25	setServerAuthority	3697
6.857.3.26	setSource	3697
6.857.3.27	setUserInfo	3697
6.857.3.28	setValid	3697
6.858	decaf::net::URL Class Reference	3697
6.858.1	Detailed Description	3698
6.858.2	Constructor & Destructor Documentation	3699
6.858.2.1	URL	3699
6.858.2.2	URL	3699
6.858.2.3	~URL	3699
6.859	decaf::net::URLDecoder Class Reference	3699
6.859.1	Constructor & Destructor Documentation	3700
6.859.1.1	~URLDecoder	3700
6.859.2	Member Function Documentation	3700
6.859.2.1	decode	3700
6.860	decaf::net::URLEncoder Class Reference	3700
6.860.1	Constructor & Destructor Documentation	3701
6.860.1.1	~URLEncoder	3701
6.860.2	Member Function Documentation	3701
6.860.2.1	encode	3701
6.861	activemq::util::Usage Class Reference	3701
6.861.1	Constructor & Destructor Documentation	3702
6.861.1.1	~Usage	3702
6.861.2	Member Function Documentation	3702
6.861.2.1	decreaseUsage	3702
6.861.2.2	enqueueUsage	3702
6.861.2.3	increaseUsage	3702
6.861.2.4	isFull	3703
6.861.2.5	waitForSpace	3703
6.861.2.6	waitForSpace	3703
6.862	decaf::io::UTFDataFormatException Class Reference	3703
6.862.1	Detailed Description	3704
6.862.2	Constructor & Destructor Documentation	3704
6.862.2.1	UTFDataFormatException	3704

6.862.2.2 UTFDataFormatException	3704
6.862.2.3 UTFDataFormatException	3704
6.862.2.4 UTFDataFormatException	3705
6.862.2.5 UTFDataFormatException	3705
6.862.2.6 UTFDataFormatException	3705
6.862.2.7 ~UTFDataFormatException	3705
6.862.3 Member Function Documentation	3705
6.862.3.1 clone	3705
6.863decaf::util::UUID Class Reference	3706
6.863.1 Detailed Description	3707
6.863.2 Constructor & Destructor Documentation	3708
6.863.2.1 UUID	3708
6.863.2.2 ~UUID	3708
6.863.3 Member Function Documentation	3708
6.863.3.1 clockSequence	3708
6.863.3.2 compareTo	3708
6.863.3.3 equals	3708
6.863.3.4 fromString	3709
6.863.3.5 getLeastSignificantBits	3709
6.863.3.6 getMostSignificantBits	3709
6.863.3.7 nameUUIDFromBytes	3709
6.863.3.8 nameUUIDFromBytes	3710
6.863.3.9 node	3710
6.863.3.10operator<	3710
6.863.3.11operator==	3711
6.863.3.12randomUUID	3711
6.863.3.13timestamp	3711
6.863.3.14toString	3711
6.863.3.15variant	3712
6.863.3.16version	3712
6.864activemq::wireformat::WireFormat Class Reference	3712
6.864.1 Detailed Description	3713
6.864.2 Constructor & Destructor Documentation	3714
6.864.2.1 ~WireFormat	3714
6.864.3 Member Function Documentation	3714
6.864.3.1 createNegotiator	3714

6.864.3.2	getVersion	3714
6.864.3.3	hasNegotiator	3714
6.864.3.4	inReceive	3715
6.864.3.5	marshal	3715
6.864.3.6	setVersion	3715
6.864.3.7	unmarshal	3716
6.865	activemq::wireformat::WireFormatFactory Class Reference	3716
6.865.1	Detailed Description	3717
6.865.2	Constructor & Destructor Documentation	3717
6.865.2.1	~WireFormatFactory	3717
6.865.3	Member Function Documentation	3717
6.865.3.1	createWireFormat	3717
6.866	activemq::commands::WireFormatInfo Class Reference	3717
6.866.1	Constructor & Destructor Documentation	3720
6.866.1.1	WireFormatInfo	3720
6.866.1.2	~WireFormatInfo	3720
6.866.2	Member Function Documentation	3720
6.866.2.1	afterUnmarshal	3720
6.866.2.2	beforeMarshal	3720
6.866.2.3	cloneDataStructure	3721
6.866.2.4	copyDataStructure	3721
6.866.2.5	equals	3721
6.866.2.6	getCacheSize	3721
6.866.2.7	getDataStructureType	3721
6.866.2.8	getMagic	3722
6.866.2.9	getMarshaledProperties	3722
6.866.2.10	getMaxInactivityDuration	3722
6.866.2.11	getMaxInactivityDurationInitialDelay	3722
6.866.2.12	getProperties	3723
6.866.2.13	getProperties	3723
6.866.2.14	getVersion	3723
6.866.2.15	isCacheEnabled	3723
6.866.2.16	isMarshalAware	3723
6.866.2.17	isSizePrefixDisabled	3724
6.866.2.18	isStackTraceEnabled	3724
6.866.2.19	isTcpNoDelayEnabled	3724

6.866.2.20	isTightEncodingEnabled	3724
6.866.2.21	isValid	3724
6.866.2.22	isWireFormatInfo	3724
6.866.2.23	setCacheEnabled	3725
6.866.2.24	setCacheSize	3725
6.866.2.25	setMagic	3725
6.866.2.26	setMarshaledProperties	3725
6.866.2.27	setMaxInactivityDuration	3725
6.866.2.28	setMaxInactivityDurationInitialDelay	3726
6.866.2.29	setProperties	3726
6.866.2.30	setSizePrefixDisabled	3726
6.866.2.31	setStackTraceEnabled	3726
6.866.2.32	setTcpNoDelayEnabled	3726
6.866.2.33	setTightEncodingEnabled	3727
6.866.2.34	setVersion	3727
6.866.2.35	toString	3727
6.866.2.36	visit	3727
6.866.3	Field Documentation	3727
6.866.3.1	ID_WIREFORMATINFO	3727
6.867	activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller Class Reference	3728
6.867.1	Detailed Description	3729
6.867.2	Constructor & Destructor Documentation	3729
6.867.2.1	WireFormatInfoMarshaller	3729
6.867.2.2	~WireFormatInfoMarshaller	3729
6.867.3	Member Function Documentation	3729
6.867.3.1	createObject	3729
6.867.3.2	getDataStructureType	3729
6.867.3.3	looseMarshal	3729
6.867.3.4	looseUnmarshal	3730
6.867.3.5	tightMarshal1	3730
6.867.3.6	tightMarshal2	3731
6.867.3.7	tightUnmarshal	3731
6.868	activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller Class Reference	3731
6.868.1	Detailed Description	3732
6.868.2	Constructor & Destructor Documentation	3733

6.868.2.1 WireFormatInfoMarshaller	3733
6.868.2.2 ~WireFormatInfoMarshaller	3733
6.868.3 Member Function Documentation	3733
6.868.3.1 createObject	3733
6.868.3.2 getDataStructureType	3733
6.868.3.3 looseMarshal	3733
6.868.3.4 looseUnmarshal	3734
6.868.3.5 tightMarshal1	3734
6.868.3.6 tightMarshal2	3734
6.868.3.7 tightUnmarshal	3735
6.869activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller Class Reference	3735
6.869.1 Detailed Description	3736
6.869.2 Constructor & Destructor Documentation	3737
6.869.2.1 WireFormatInfoMarshaller	3737
6.869.2.2 ~WireFormatInfoMarshaller	3737
6.869.3 Member Function Documentation	3737
6.869.3.1 createObject	3737
6.869.3.2 getDataStructureType	3737
6.869.3.3 looseMarshal	3737
6.869.3.4 looseUnmarshal	3738
6.869.3.5 tightMarshal1	3738
6.869.3.6 tightMarshal2	3738
6.869.3.7 tightUnmarshal	3739
6.870activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller Class Reference	3739
6.870.1 Detailed Description	3740
6.870.2 Constructor & Destructor Documentation	3741
6.870.2.1 WireFormatInfoMarshaller	3741
6.870.2.2 ~WireFormatInfoMarshaller	3741
6.870.3 Member Function Documentation	3741
6.870.3.1 createObject	3741
6.870.3.2 getDataStructureType	3741
6.870.3.3 looseMarshal	3741
6.870.3.4 looseUnmarshal	3742
6.870.3.5 tightMarshal1	3742
6.870.3.6 tightMarshal2	3742

6.870.3.7 tightUnmarshal	3743
6.871activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller Class Reference	3743
6.871.1 Detailed Description	3744
6.871.2 Constructor & Destructor Documentation	3745
6.871.2.1 WireFormatInfoMarshaller	3745
6.871.2.2 ~WireFormatInfoMarshaller	3745
6.871.3 Member Function Documentation	3745
6.871.3.1 createObject	3745
6.871.3.2 getDataStructureType	3745
6.871.3.3 looseMarshal	3745
6.871.3.4 looseUnmarshal	3746
6.871.3.5 tightMarshal1	3746
6.871.3.6 tightMarshal2	3746
6.871.3.7 tightUnmarshal	3747
6.872activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller Class Reference	3747
6.872.1 Detailed Description	3748
6.872.2 Constructor & Destructor Documentation	3749
6.872.2.1 WireFormatInfoMarshaller	3749
6.872.2.2 ~WireFormatInfoMarshaller	3749
6.872.3 Member Function Documentation	3749
6.872.3.1 createObject	3749
6.872.3.2 getDataStructureType	3749
6.872.3.3 looseMarshal	3749
6.872.3.4 looseUnmarshal	3750
6.872.3.5 tightMarshal1	3750
6.872.3.6 tightMarshal2	3750
6.872.3.7 tightUnmarshal	3751
6.873activemq::wireformat::WireFormatNegotiator Class Reference	3751
6.873.1 Detailed Description	3752
6.873.2 Constructor & Destructor Documentation	3752
6.873.2.1 WireFormatNegotiator	3752
6.873.2.2 ~WireFormatNegotiator	3752
6.874activemq::wireformat::WireFormatRegistry Class Reference	3752
6.874.1 Detailed Description	3753
6.874.2 Constructor & Destructor Documentation	3753

6.874.2.1 ~WireFormatRegistry	3753
6.874.3 Member Function Documentation	3753
6.874.3.1 findFactory	3753
6.874.3.2 getInstance	3754
6.874.3.3 getWireFormatNames	3754
6.874.3.4 registerFactory	3754
6.874.3.5 unregisterFactory	3754
6.875activemq::transport::inactivity::WriteChecker Class Reference	3755
6.875.1 Detailed Description	3755
6.875.2 Constructor & Destructor Documentation	3755
6.875.2.1 WriteChecker	3755
6.875.2.2 ~WriteChecker	3755
6.875.3 Member Function Documentation	3755
6.875.3.1 run	3755
6.876decaf::io::Writer Class Reference	3756
6.876.1 Constructor & Destructor Documentation	3757
6.876.1.1 Writer	3757
6.876.1.2 ~Writer	3757
6.876.2 Member Function Documentation	3757
6.876.2.1 append	3757
6.876.2.2 append	3758
6.876.2.3 append	3758
6.876.2.4 doAppendChar	3759
6.876.2.5 doAppendCharSequence	3759
6.876.2.6 doAppendCharSequenceStartEnd	3759
6.876.2.7 doWriteArray	3759
6.876.2.8 doWriteArrayBounded	3759
6.876.2.9 doWriteChar	3759
6.876.2.10doWriteString	3759
6.876.2.11doWriteStringBounded	3759
6.876.2.12doWriteVector	3759
6.876.2.13write	3759
6.876.2.14write	3760
6.876.2.15write	3760
6.876.2.16write	3760
6.876.2.17write	3761

6.876.2.18	write	3761
6.877	decaf::security::auth::x500::X500Principal Class Reference	3761
6.877.1	Constructor & Destructor Documentation	3762
6.877.1.1	~X500Principal	3762
6.877.2	Member Function Documentation	3762
6.877.2.1	getEncoded	3762
6.877.2.2	getName	3762
6.877.2.3	hashCode	3762
6.878	decaf::security::cert::X509Certificate Class Reference	3762
6.878.1	Detailed Description	3763
6.878.2	Constructor & Destructor Documentation	3764
6.878.2.1	~X509Certificate	3764
6.878.3	Member Function Documentation	3764
6.878.3.1	checkValidity	3764
6.878.3.2	checkValidity	3764
6.878.3.3	getBasicConstraints	3764
6.878.3.4	getIssuerUniqueID	3764
6.878.3.5	getIssuerX500Principal	3764
6.878.3.6	getKeyUsage	3764
6.878.3.7	getNotAfter	3764
6.878.3.8	getNotBefore	3764
6.878.3.9	getSigAlgName	3764
6.878.3.10	getSigAlgOID	3764
6.878.3.11	getSigAlgParams	3764
6.878.3.12	getSignature	3764
6.878.3.13	getSubjectUniqueID	3764
6.878.3.14	getSubjectX500Principal	3764
6.878.3.15	getTBSCertificate	3764
6.878.3.16	getVersion	3764
6.879	activemq::commands::XATransactionId Class Reference	3765
6.879.1	Member Typedef Documentation	3766
6.879.1.1	COMPARATOR	3766
6.879.2	Constructor & Destructor Documentation	3766
6.879.2.1	XATransactionId	3766
6.879.2.2	XATransactionId	3766
6.879.2.3	~XATransactionId	3766

6.879.3 Member Function Documentation	3766
6.879.3.1 cloneDataStructure	3766
6.879.3.2 compareTo	3766
6.879.3.3 copyDataStructure	3766
6.879.3.4 equals	3767
6.879.3.5 equals	3767
6.879.3.6 getBranchQualifier	3767
6.879.3.7 getBranchQualifier	3767
6.879.3.8 getDataStructureType	3767
6.879.3.9 getFormatId	3768
6.879.3.10 getGlobalTransactionId	3768
6.879.3.11 getGlobalTransactionId	3768
6.879.3.12 operator<	3768
6.879.3.13 operator=	3768
6.879.3.14 operator==	3768
6.879.3.15 setBranchQualifier	3768
6.879.3.16 setFormatId	3768
6.879.3.17 setGlobalTransactionId	3768
6.879.3.18 toString	3768
6.879.4 Field Documentation	3769
6.879.4.1 branchQualifier	3769
6.879.4.2 formatId	3769
6.879.4.3 globalTransactionId	3769
6.879.4.4 ID_XATRANSACTIONID	3769
6.880 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller Class	
Reference	3769
6.880.1 Detailed Description	3770
6.880.2 Constructor & Destructor Documentation	3770
6.880.2.1 XATransactionIdMarshaller	3770
6.880.2.2 ~XATransactionIdMarshaller	3770
6.880.3 Member Function Documentation	3770
6.880.3.1 createObject	3770
6.880.3.2 getDataStructureType	3770
6.880.3.3 looseMarshal	3771
6.880.3.4 looseUnmarshal	3771
6.880.3.5 tightMarshal1	3771
6.880.3.6 tightMarshal2	3772

6.880.3.7 tightUnmarshal	3772
6.881activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller Class	
Reference	3773
6.881.1 Detailed Description	3774
6.881.2 Constructor & Destructor Documentation	3774
6.881.2.1 XATransactionIdMarshaller	3774
6.881.2.2 ~XATransactionIdMarshaller	3774
6.881.3 Member Function Documentation	3774
6.881.3.1 createObject	3774
6.881.3.2 getDataStructureType	3774
6.881.3.3 looseMarshal	3775
6.881.3.4 looseUnmarshal	3775
6.881.3.5 tightMarshal1	3775
6.881.3.6 tightMarshal2	3776
6.881.3.7 tightUnmarshal	3776
6.882activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller Class	
Reference	3777
6.882.1 Detailed Description	3778
6.882.2 Constructor & Destructor Documentation	3778
6.882.2.1 XATransactionIdMarshaller	3778
6.882.2.2 ~XATransactionIdMarshaller	3778
6.882.3 Member Function Documentation	3778
6.882.3.1 createObject	3778
6.882.3.2 getDataStructureType	3778
6.882.3.3 looseMarshal	3779
6.882.3.4 looseUnmarshal	3779
6.882.3.5 tightMarshal1	3779
6.882.3.6 tightMarshal2	3780
6.882.3.7 tightUnmarshal	3780
6.883activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller Class	
Reference	3781
6.883.1 Detailed Description	3782
6.883.2 Constructor & Destructor Documentation	3782
6.883.2.1 XATransactionIdMarshaller	3782
6.883.2.2 ~XATransactionIdMarshaller	3782
6.883.3 Member Function Documentation	3782
6.883.3.1 createObject	3782

6.883.3.2	getDataStructureType	3782
6.883.3.3	looseMarshal	3783
6.883.3.4	looseUnmarshal	3783
6.883.3.5	tightMarshal1	3783
6.883.3.6	tightMarshal2	3784
6.883.3.7	tightUnmarshal	3784
6.884	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller Class	
	Reference	3785
6.884.1	Detailed Description	3786
6.884.2	Constructor & Destructor Documentation	3786
6.884.2.1	XATransactionIdMarshaller	3786
6.884.2.2	~XATransactionIdMarshaller	3786
6.884.3	Member Function Documentation	3786
6.884.3.1	createObject	3786
6.884.3.2	getDataStructureType	3786
6.884.3.3	looseMarshal	3787
6.884.3.4	looseUnmarshal	3787
6.884.3.5	tightMarshal1	3787
6.884.3.6	tightMarshal2	3788
6.884.3.7	tightUnmarshal	3788
6.885	activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller Class	
	Reference	3789
6.885.1	Detailed Description	3790
6.885.2	Constructor & Destructor Documentation	3790
6.885.2.1	XATransactionIdMarshaller	3790
6.885.2.2	~XATransactionIdMarshaller	3790
6.885.3	Member Function Documentation	3790
6.885.3.1	createObject	3790
6.885.3.2	getDataStructureType	3790
6.885.3.3	looseMarshal	3791
6.885.3.4	looseUnmarshal	3791
6.885.3.5	tightMarshal1	3791
6.885.3.6	tightMarshal2	3792
6.885.3.7	tightUnmarshal	3792
6.886	decaf::util::logging::XMLFormatter Class Reference	3793
6.886.1	Detailed Description	3793
6.886.2	Constructor & Destructor Documentation	3794

6.886.2.1 XMLFormatter	3794
6.886.2.2 ~XMLFormatter	3794
6.886.3 Member Function Documentation	3794
6.886.3.1 format	3794
6.886.3.2 getHead	3794
6.886.3.3 getTail	3794
6.887z_stream_s Struct Reference	3795
6.887.1 Field Documentation	3795
6.887.1.1 Adler	3795
6.887.1.2 avail_in	3795
6.887.1.3 avail_out	3795
6.887.1.4 data_type	3795
6.887.1.5 msg	3795
6.887.1.6 next_in	3795
6.887.1.7 next_out	3795
6.887.1.8 opaque	3795
6.887.1.9 reserved	3795
6.887.1.10 state	3795
6.887.1.11 total_in	3795
6.887.1.12 total_out	3795
6.887.1.13 alloc	3795
6.887.1.14 free	3795
6.888deflate::util::zip::ZipException Class Reference	3796
6.888.1 Constructor & Destructor Documentation	3796
6.888.1.1 ZipException	3796
6.888.1.2 ZipException	3796
6.888.1.3 ZipException	3797
6.888.1.4 ZipException	3797
6.888.1.5 ZipException	3797
6.888.1.6 ZipException	3797
6.888.1.7 ~ZipException	3798
6.888.2 Member Function Documentation	3798
6.888.2.1 clone	3798
7 File Documentation	3799
7.1 src/main/activemq/cmsutil/CachedConsumer.h File Reference	3799
7.2 src/main/activemq/cmsutil/CachedProducer.h File Reference	3799

7.3	src/main/activemq/cmsutil/CmsAccessor.h File Reference	3800
7.4	src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference	3800
7.5	src/main/activemq/cmsutil/CmsTemplate.h File Reference	3801
7.6	src/main/activemq/cmsutil/DestinationResolver.h File Reference	3801
7.7	src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference	3802
7.8	src/main/activemq/cmsutil/MessageCreator.h File Reference	3802
7.9	src/main/activemq/cmsutil/PooledSession.h File Reference	3803
7.10	src/main/activemq/cmsutil/ProducerCallback.h File Reference	3803
7.11	src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference	3804
7.12	src/main/decaf/internal/util/ResourceLifecycleManager.h File Reference	3805
7.13	src/main/activemq/cmsutil/SessionCallback.h File Reference	3805
7.14	src/main/activemq/cmsutil/SessionPool.h File Reference	3805
7.15	src/main/activemq/commands/ActiveMQBlobMessage.h File Reference	3806
7.16	src/main/activemq/commands/ActiveMQBytesMessage.h File Reference	3806
7.17	src/main/activemq/commands/ActiveMQDestination.h File Reference	3807
7.18	src/main/activemq/commands/ActiveMQMapMessage.h File Reference	3808
7.19	src/main/activemq/commands/ActiveMQMessage.h File Reference	3808
7.20	src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference	3809
7.21	src/main/activemq/commands/ActiveMQObjectMessage.h File Reference	3809
7.22	src/main/activemq/commands/ActiveMQQueue.h File Reference	3810
7.23	src/main/activemq/commands/ActiveMQStreamMessage.h File Reference	3810
7.24	src/main/activemq/commands/ActiveMQTempDestination.h File Reference	3811
7.25	src/main/activemq/commands/ActiveMQTempQueue.h File Reference	3812
7.26	src/main/activemq/commands/ActiveMQTempTopic.h File Reference	3812
7.27	src/main/activemq/commands/ActiveMQTextMessage.h File Reference	3813
7.28	src/main/activemq/commands/ActiveMQTopic.h File Reference	3813
7.29	src/main/activemq/commands/BaseCommand.h File Reference	3814
7.30	src/main/activemq/commands/BaseDataStructure.h File Reference	3814
7.31	src/main/activemq/commands/BooleanExpression.h File Reference	3815
7.32	src/main/activemq/commands/BrokerError.h File Reference	3815
7.33	src/main/activemq/commands/BrokerId.h File Reference	3816
7.34	src/main/activemq/commands/BrokerInfo.h File Reference	3816
7.35	src/main/activemq/commands/Command.h File Reference	3817
7.36	src/main/activemq/commands/ConnectionControl.h File Reference	3817
7.37	src/main/activemq/commands/ConnectionError.h File Reference	3818
7.38	src/main/activemq/commands/ConnectionId.h File Reference	3818

7.39	src/main/activemq/commands/ConnectionInfo.h File Reference	3819
7.40	src/main/activemq/commands/ConsumerControl.h File Reference	3819
7.41	src/main/activemq/commands/ConsumerId.h File Reference	3820
7.42	src/main/activemq/commands/ConsumerInfo.h File Reference	3821
7.43	src/main/activemq/commands/ControlCommand.h File Reference	3821
7.44	src/main/activemq/commands/DataArrayResponse.h File Reference	3822
7.45	src/main/activemq/commands/DataResponse.h File Reference	3822
7.46	src/main/activemq/commands/DataStructure.h File Reference	3823
7.47	src/main/activemq/commands/DestinationInfo.h File Reference	3823
7.48	src/main/activemq/commands/DiscoveryEvent.h File Reference	3824
7.49	src/main/activemq/commands/ExceptionResponse.h File Reference	3824
7.50	src/main/activemq/commands/FlushCommand.h File Reference	3825
7.51	src/main/activemq/commands/IntegerResponse.h File Reference	3825
7.52	src/main/activemq/commands/JournalQueueAck.h File Reference	3826
7.53	src/main/activemq/commands/JournalTopicAck.h File Reference	3826
7.54	src/main/activemq/commands/JournalTrace.h File Reference	3827
7.55	src/main/activemq/commands/JournalTransaction.h File Reference	3827
7.56	src/main/activemq/commands/KeepAliveInfo.h File Reference	3828
7.57	src/main/activemq/commands/LastPartialCommand.h File Reference	3828
7.58	src/main/activemq/commands/LocalTransactionId.h File Reference	3829
7.59	src/main/activemq/commands/Message.h File Reference	3829
7.60	src/main/cms/Message.h File Reference	3830
7.61	src/main/activemq/commands/MessageAck.h File Reference	3831
7.62	src/main/activemq/commands/MessageDispatch.h File Reference	3831
7.63	src/main/activemq/commands/MessageDispatchNotification.h File Reference . . .	3832
7.64	src/main/activemq/commands/MessageId.h File Reference	3832
7.65	src/main/activemq/commands/MessagePull.h File Reference	3833
7.66	src/main/activemq/commands/NetworkBridgeFilter.h File Reference	3834
7.67	src/main/activemq/commands/PartialCommand.h File Reference	3834
7.68	src/main/activemq/commands/ProducerAck.h File Reference	3835
7.69	src/main/activemq/commands/ProducerId.h File Reference	3835
7.70	src/main/activemq/commands/ProducerInfo.h File Reference	3836
7.71	src/main/activemq/commands/RemoveInfo.h File Reference	3836
7.72	src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference	3837
7.73	src/main/activemq/commands/ReplayCommand.h File Reference	3837
7.74	src/main/activemq/commands/Response.h File Reference	3838

7.75	src/main/activemq/commands/SessionId.h File Reference	3838
7.76	src/main/activemq/commands/SessionInfo.h File Reference	3839
7.77	src/main/activemq/commands/ShutdownInfo.h File Reference	3839
7.78	src/main/activemq/commands/SubscriptionInfo.h File Reference	3840
7.79	src/main/activemq/commands/TransactionId.h File Reference	3840
7.80	src/main/activemq/commands/TransactionInfo.h File Reference	3841
7.81	src/main/activemq/commands/WireFormatInfo.h File Reference	3841
7.82	src/main/activemq/commands/XATransactionId.h File Reference	3842
7.83	src/main/activemq/core/ActiveMQAckHandler.h File Reference	3842
7.84	src/main/activemq/core/ActiveMQConnection.h File Reference	3843
7.85	src/main/activemq/core/ActiveMQConnectionFactory.h File Reference	3844
7.86	src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference	3844
7.87	src/main/activemq/core/ActiveMQConstants.h File Reference	3845
7.88	src/main/activemq/core/ActiveMQConsumer.h File Reference	3845
7.89	src/main/activemq/core/ActiveMQProducer.h File Reference	3846
7.90	src/main/activemq/core/ActiveMQQueueBrowser.h File Reference	3847
7.91	src/main/activemq/core/ActiveMQSession.h File Reference	3847
7.92	src/main/activemq/core/ActiveMQSessionExecutor.h File Reference	3848
7.93	src/main/activemq/core/ActiveMQTransactionContext.h File Reference	3849
7.94	src/main/activemq/core/DispatchData.h File Reference	3849
7.95	src/main/activemq/core/Dispatcher.h File Reference	3850
7.96	src/main/activemq/core/MessageDispatchChannel.h File Reference	3850
7.97	src/main/activemq/core/policies/DefaultPrefetchPolicy.h File Reference	3851
7.98	src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference	3851
7.99	src/main/activemq/core/PrefetchPolicy.h File Reference	3852
7.100	src/main/activemq/core/RedeliveryPolicy.h File Reference	3852
7.101	src/main/activemq/core/Synchronization.h File Reference	3853
7.102	src/main/activemq/exceptions/ActiveMQException.h File Reference	3853
7.103	src/main/activemq/exceptions/BrokerException.h File Reference	3854
7.104	src/main/activemq/exceptions/ExceptionDefines.h File Reference	3854
7.104.1	Define Documentation	3855
7.104.1.1	AMQ_CATCH_EXCEPTION_CONVERT	3855
7.104.1.2	AMQ_CATCH_NOTHROW	3855
7.104.1.3	AMQ_CATCH_RETHROW	3855
7.104.1.4	AMQ_CATCHALL_NOTHROW	3856
7.104.1.5	AMQ_CATCHALL_THROW	3856

7.105src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference	3856
7.105.1 Define Documentation	3857
7.105.1.1 DECAF_CATCH_EXCEPTION_CONVERT	3857
7.105.1.2 DECAF_CATCH_NOTHROW	3857
7.105.1.3 DECAF_CATCH_RETHROW	3857
7.105.1.4 DECAF_CATCHALL_NOTHROW	3858
7.105.1.5 DECAF_CATCHALL_THROW	3858
7.106src/main/activemq/io/LoggingInputStream.h File Reference	3858
7.107src/main/activemq/io/LoggingOutputStream.h File Reference	3859
7.108src/main/activemq/library/ActiveMQCPP.h File Reference	3859
7.109src/main/activemq/state/CommandVisitor.h File Reference	3860
7.110src/main/activemq/state/CommandVisitorAdapter.h File Reference	3860
7.111src/main/activemq/state/ConnectionState.h File Reference	3861
7.112src/main/activemq/state/ConnectionStateTracker.h File Reference	3862
7.113src/main/activemq/state/ConsumerState.h File Reference	3863
7.114src/main/activemq/state/ProducerState.h File Reference	3863
7.115src/main/activemq/state/SessionState.h File Reference	3864
7.116src/main/activemq/state/Tracked.h File Reference	3865
7.117src/main/activemq/state/TransactionState.h File Reference	3865
7.118src/main/activemq/threads/CompositeTask.h File Reference	3866
7.119src/main/activemq/threads/CompositeTaskRunner.h File Reference	3866
7.120src/main/activemq/threads/DedicatedTaskRunner.h File Reference	3867
7.121src/main/activemq/threads/Task.h File Reference	3867
7.122src/main/activemq/threads/TaskRunner.h File Reference	3868
7.123src/main/activemq/transport/AbstractTransportFactory.h File Reference	3868
7.124src/main/activemq/transport/CompositeTransport.h File Reference	3869
7.125src/main/activemq/transport/correlator/FutureResponse.h File Reference	3869
7.126src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference	3870
7.127src/main/activemq/transport/DefaultTransportListener.h File Reference	3871
7.128src/main/activemq/transport/failover/BackupTransport.h File Reference	3871
7.129src/main/activemq/transport/failover/BackupTransportPool.h File Reference	3872
7.130src/main/activemq/transport/failover/CloseTransportsTask.h File Reference	3872
7.131src/main/activemq/transport/failover/FailoverTransport.h File Reference	3873
7.132src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference	3874
7.133src/main/activemq/transport/failover/FailoverTransportListener.h File Reference	3874
7.134src/main/activemq/transport/failover/URIPool.h File Reference	3875

7.135src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference	3875
7.136src/main/activemq/transport/inactivity/ReadChecker.h File Reference	3876
7.137src/main/activemq/transport/inactivity/WriteChecker.h File Reference	3876
7.138src/main/activemq/transport/IOTransport.h File Reference	3877
7.139src/main/activemq/transport/logging/LoggingTransport.h File Reference	3878
7.140src/main/activemq/transport/mock/InternalCommandListener.h File Reference	3878
7.141src/main/activemq/transport/mock/MockTransport.h File Reference	3879
7.142src/main/activemq/transport/mock/MockTransportFactory.h File Reference	3880
7.143src/main/activemq/transport/mock/ResponseBuilder.h File Reference	3880
7.144src/main/activemq/transport/tcp/SslTransport.h File Reference	3881
7.145src/main/activemq/transport/tcp/SslTransportFactory.h File Reference	3881
7.146src/main/activemq/transport/tcp/TcpTransport.h File Reference	3882
7.147src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference	3882
7.148src/main/activemq/transport/Transport.h File Reference	3883
7.149src/main/activemq/transport/TransportFactory.h File Reference	3884
7.150src/main/activemq/transport/TransportFilter.h File Reference	3884
7.151src/main/activemq/transport/TransportListener.h File Reference	3885
7.152src/main/activemq/transport/TransportRegistry.h File Reference	3885
7.153src/main/activemq/util/ActiveMQProperties.h File Reference	3886
7.154src/main/activemq/util/CMSExceptionSupport.h File Reference	3886
7.154.1 Define Documentation	3887
7.154.1.1 AMQ_CATCH_ALL_THROW_CMSEXCEPTION	3887
7.155src/main/activemq/util/CompositeData.h File Reference	3888
7.156src/main/activemq/util/Config.h File Reference	3889
7.156.1 Define Documentation	3889
7.156.1.1 AMQCPP_API	3889
7.156.1.2 HAVE_PTHREAD_H	3889
7.156.1.3 HAVE_UUID_T	3889
7.156.1.4 HAVE_UUID_UUID_H	3889
7.157src/main/cms/Config.h File Reference	3889
7.157.1 Define Documentation	3889
7.157.1.1 CMS_API	3889
7.158src/main/decaf/util/Config.h File Reference	3889
7.158.1 Define Documentation	3890
7.158.1.1 DECAF_API	3890
7.158.1.2 DECAF_UNUSED	3890

7.158.1.3 HAVE_PTHREAD_H	3890
7.158.1.4 HAVE_UUID_T	3890
7.158.1.5 HAVE_UUID_UUID_H	3890
7.159src/main/activemq/util/IdGenerator.h File Reference	3890
7.160src/main/activemq/util/LongSequenceGenerator.h File Reference	3890
7.161src/main/activemq/util/MarshallingSupport.h File Reference	3891
7.162src/main/activemq/util/MemoryUsage.h File Reference	3891
7.163src/main/activemq/util/PrimitiveList.h File Reference	3892
7.164src/main/activemq/util/PrimitiveMap.h File Reference	3892
7.165src/main/activemq/util/PrimitiveValueConverter.h File Reference	3893
7.166src/main/activemq/util/PrimitiveValueNode.h File Reference	3893
7.167src/main/activemq/util/URISupport.h File Reference	3894
7.168src/main/activemq/util/Usage.h File Reference	3894
7.169src/main/activemq/wireformat/MarshalAware.h File Reference	3895
7.170src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h File Reference	3895
7.171src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h File Reference	3896
7.172src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference	3897
7.173src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h File Reference	3897
7.174src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h File Reference	3898
7.175src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h File Reference	3899
7.176src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h File Reference	3899
7.177src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h File Reference	3900
7.178src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarshaller.h File Reference	3901
7.179src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h File Reference	3901
7.180src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h File Reference	3902
7.181src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File Reference	3903
7.182src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h File Reference	3903

7.183	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h	
	File Reference	3904
7.184	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMessageMarshaller.h	
	File Reference	3905
7.185	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h	
	File Reference	3905
7.186	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h	
	File Reference	3906
7.187	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h	
	File Reference	3907
7.188	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h	
	File Reference	3907
7.189	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h	
	File Reference	3908
7.190	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h	
	File Reference	3909
7.191	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h	
	File Reference	3909
7.192	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h	
	File Reference	3910
7.193	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h	
	File Reference	3911
7.194	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h	
	File Reference	3911
7.195	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h	
	File Reference	3912
7.196	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMessageMarshaller.h	
	File Reference	3913
7.197	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h	
	File Reference	3913
7.198	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h	
	File Reference	3914
7.199	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h	
	File Reference	3915
7.200	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h	
	File Reference	3915
7.201	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h	
	File Reference	3916
7.202	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h	
	File Reference	3917
7.203	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h	
	File Reference	3917
7.204	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h	
	File Reference	3918

7.205	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h	
	File Reference	3919
7.206	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h	
	File Reference	3919
7.207	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h	
	File Reference	3920
7.208	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMarshaller.h	
	File Reference	3921
7.209	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h	
	File Reference	3921
7.210	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h	
	File Reference	3922
7.211	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h	
	File Reference	3923
7.212	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h	
	File Reference	3923
7.213	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h	
	File Reference	3924
7.214	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h	
	File Reference	3925
7.215	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h	
	File Reference	3925
7.216	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h	
	File Reference	3926
7.217	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h	
	File Reference	3927
7.218	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h	
	File Reference	3927
7.219	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h	
	File Reference	3928
7.220	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamMessageMarshaller.h	
	File Reference	3929
7.221	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h	
	File Reference	3929
7.222	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h	
	File Reference	3930
7.223	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h	
	File Reference	3931
7.224	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h	
	File Reference	3931
7.225	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h	
	File Reference	3932
7.226	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h	
	File Reference	3933

7.227	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h	
	File Reference	3933
7.228	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h	
	File Reference	3934
7.229	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h	
	File Reference	3935
7.230	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h	
	File Reference	3935
7.231	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h	
	File Reference	3936
7.232	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarshaller.h	
	File Reference	3937
7.233	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h	
	File Reference	3937
7.234	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h	
	File Reference	3938
7.235	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h	
	File Reference	3939
7.236	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h	
	File Reference	3939
7.237	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h	
	File Reference	3940
7.238	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h	
	File Reference	3941
7.239	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h	
	File Reference	3941
7.240	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h	
	File Reference	3942
7.241	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h	
	File Reference	3943
7.242	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h	
	File Reference	3943
7.243	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h	
	File Reference	3944
7.244	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMessageMarshaller.h	
	File Reference	3945
7.245	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h	
	File Reference	3945
7.246	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h	
	File Reference	3946
7.247	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h	
	File Reference	3947
7.248	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h	
	File Reference	3947

7.249src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h	
File Reference	3948
7.250src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h	
File Reference	3949
7.251src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h	
File Reference	3949
7.252src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h	
File Reference	3950
7.253src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h	
File Reference	3951
7.254src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h	
File Reference	3951
7.255src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h	
File Reference	3952
7.256src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h	
File Reference	3953
7.257src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h	File
Reference	3953
7.258src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h	File
Reference	3954
7.259src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h	File
Reference	3955
7.260src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h	File
Reference	3955
7.261src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h	File
Reference	3956
7.262src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h	File
Reference	3957
7.263src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h	File
Reference	3957
7.264src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h	File
Reference	3958
7.265src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h	File
Reference	3959
7.266src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h	File
Reference	3959
7.267src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h	File
Reference	3960
7.268src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h	File
Reference	3961
7.269src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h	
File Reference	3961
7.270src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h	
File Reference	3962

7.271	src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h	
	File Reference	3963
7.272	src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h	
	File Reference	3963
7.273	src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h	
	File Reference	3964
7.274	src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h	
	File Reference	3965
7.275	src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h	
	File Reference	3965
7.276	src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h	
	File Reference	3966
7.277	src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h	
	File Reference	3967
7.278	src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h	
	File Reference	3967
7.279	src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h	
	File Reference	3968
7.280	src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h	
	File Reference	3969
7.281	src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h	
	File Reference	3969
7.282	src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h	
	File Reference	3970
7.283	src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h	
	File Reference	3971
7.284	src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h	
	File Reference	3971
7.285	src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h	
	File Reference	3972
7.286	src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h	
	File Reference	3973
7.287	src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h	
	File Reference	3973
7.288	src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h	
	File Reference	3974
7.289	src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h	
	File Reference	3975
7.290	src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h	
	File Reference	3975
7.291	src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h	
	File Reference	3976
7.292	src/main/activemq/wireformat/openwire/marshal/v6/ConnectionInfoMarshaller.h	
	File Reference	3977

7.293	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h	
	File Reference	3977
7.294	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h	
	File Reference	3978
7.295	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h	
	File Reference	3979
7.296	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h	
	File Reference	3979
7.297	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h	
	File Reference	3980
7.298	src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h	
	File Reference	3981
7.299	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h	
	File Reference	3981
7.300	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h	
	File Reference	3982
7.301	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h	
	File Reference	3983
7.302	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h	
	File Reference	3983
7.303	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h	
	File Reference	3984
7.304	src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h	
	File Reference	3985
7.305	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h	
	File Reference	3985
7.306	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h	
	File Reference	3986
7.307	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h	
	File Reference	3987
7.308	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h	
	File Reference	3987
7.309	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h	
	File Reference	3988
7.310	src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h	
	File Reference	3989
7.311	src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h	
	File Reference	3989
7.312	src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h	
	File Reference	3990
7.313	src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h	
	File Reference	3991
7.314	src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h	
	File Reference	3991

7.315src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h	
File Reference	3992
7.316src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller.h	
File Reference	3993
7.317src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h	
File Reference	3993
7.318src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h	
File Reference	3994
7.319src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h	
File Reference	3995
7.320src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h	
File Reference	3995
7.321src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h	
File Reference	3996
7.322src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h	
File Reference	3997
7.323src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h	
File Reference	3997
7.324src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h	
File Reference	3998
7.325src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h	
File Reference	3999
7.326src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h	
File Reference	3999
7.327src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h	
File Reference	4000
7.328src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h	
File Reference	4001
7.329src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h	
File Reference	4001
7.330src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h	
File Reference	4002
7.331src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h	
File Reference	4003
7.332src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h	
File Reference	4003
7.333src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h	
File Reference	4004
7.334src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h	
File Reference	4005
7.335src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h	
File Reference	4005
7.336src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h	
File Reference	4006

7.337src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h	
File Reference	4007
7.338src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h	
File Reference	4007
7.339src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h	
File Reference	4008
7.340src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller.h	
File Reference	4009
7.341src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h	
File Reference	4009
7.342src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h	
File Reference	4010
7.343src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h	
File Reference	4011
7.344src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h	
File Reference	4011
7.345src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h	
File Reference	4012
7.346src/main/activemq/wireformat/openwire/marshal/v6/ExceptionResponseMarshaller.h	
File Reference	4013
7.347src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h	
File Reference	4013
7.348src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h	
File Reference	4014
7.349src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h	
File Reference	4015
7.350src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h	
File Reference	4015
7.351src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h	
File Reference	4016
7.352src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h	
File Reference	4017
7.353src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h	
File Reference	4017
7.354src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h	
File Reference	4018
7.355src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h	
File Reference	4019
7.356src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h	
File Reference	4019
7.357src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h	
File Reference	4020
7.358src/main/activemq/wireformat/openwire/marshal/v6/IntegerResponseMarshaller.h	
File Reference	4021

7.359src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h	
File Reference	4021
7.360src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h	
File Reference	4022
7.361src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h	
File Reference	4023
7.362src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h	
File Reference	4023
7.363src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h	
File Reference	4024
7.364src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAckMarshaller.h	
File Reference	4025
7.365src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h	
File Reference	4025
7.366src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h	
File Reference	4026
7.367src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h	
File Reference	4027
7.368src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h	
File Reference	4027
7.369src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h	
File Reference	4028
7.370src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h	
File Reference	4029
7.371src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h	
File Reference	4029
7.372src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h	
File Reference	4030
7.373src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h	
File Reference	4031
7.374src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h	
File Reference	4031
7.375src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h	
File Reference	4032
7.376src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h	
File Reference	4033
7.377src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h	
File Reference	4033
7.378src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h	
File Reference	4034
7.379src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h	
File Reference	4035
7.380src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h	
File Reference	4035

7.381	src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h	
	File Reference	4036
7.382	src/main/activemq/wireformat/openwire/marshal/v6/JournalTransactionMarshaller.h	
	File Reference	4037
7.383	src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h	
	File Reference	4037
7.384	src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h	
	File Reference	4038
7.385	src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h	
	File Reference	4039
7.386	src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h	
	File Reference	4039
7.387	src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h	
	File Reference	4040
7.388	src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h	
	File Reference	4041
7.389	src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h	
	File Reference	4041
7.390	src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h	
	File Reference	4042
7.391	src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h	
	File Reference	4043
7.392	src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h	
	File Reference	4043
7.393	src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h	
	File Reference	4044
7.394	src/main/activemq/wireformat/openwire/marshal/v6/LastPartialCommandMarshaller.h	
	File Reference	4045
7.395	src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h	
	File Reference	4045
7.396	src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h	
	File Reference	4046
7.397	src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h	
	File Reference	4047
7.398	src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h	
	File Reference	4047
7.399	src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h	
	File Reference	4048
7.400	src/main/activemq/wireformat/openwire/marshal/v6/LocalTransactionIdMarshaller.h	
	File Reference	4049
7.401	src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h	File
	Reference	4049
7.402	src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h	File
	Reference	4050

7.403	src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h	File	
	Reference		4050
7.404	src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h	File	
	Reference		4051
7.405	src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h	File	
	Reference		4051
7.406	src/main/activemq/wireformat/openwire/marshal/v6/MarshallerFactory.h	File	
	Reference		4052
7.407	src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h		
	File Reference		4052
7.408	src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h		
	File Reference		4053
7.409	src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h		
	File Reference		4053
7.410	src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h		
	File Reference		4054
7.411	src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h		
	File Reference		4055
7.412	src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h		
	File Reference		4055
7.413	src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h		
	File Reference		4056
7.414	src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h		
	File Reference		4057
7.415	src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h		
	File Reference		4057
7.416	src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h		
	File Reference		4058
7.417	src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h		
	File Reference		4059
7.418	src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h		
	File Reference		4059
7.419	src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h		
	File Reference		4060
7.420	src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h		
	File Reference		4061
7.421	src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h		
	File Reference		4062
7.422	src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h		
	File Reference		4062
7.423	src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h		
	File Reference		4063
7.424	src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchNotificationMarshaller.h		
	File Reference		4064

7.425	src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h	File	
	Reference		4064
7.426	src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h	File	
	Reference		4065
7.427	src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h	File	
	Reference		4066
7.428	src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h	File	
	Reference		4066
7.429	src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h	File	
	Reference		4067
7.430	src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h	File	
	Reference		4068
7.431	src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h	File	
	Reference		4068
7.432	src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h	File	
	Reference		4069
7.433	src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h	File	
	Reference		4070
7.434	src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h	File	
	Reference		4070
7.435	src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h	File	
	Reference		4071
7.436	src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h	File	
	Reference		4072
7.437	src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h		
	File Reference		4072
7.438	src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h		
	File Reference		4073
7.439	src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h		
	File Reference		4074
7.440	src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h		
	File Reference		4074
7.441	src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h		
	File Reference		4075
7.442	src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h		
	File Reference		4076
7.443	src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h		
	File Reference		4076
7.444	src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h		
	File Reference		4077
7.445	src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h		
	File Reference		4078
7.446	src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h		
	File Reference		4078

7.447	src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h	
	File Reference	4079
7.448	src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h	
	File Reference	4080
7.449	src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h	
	File Reference	4080
7.450	src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h	
	File Reference	4081
7.451	src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h	
	File Reference	4082
7.452	src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h	
	File Reference	4082
7.453	src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h	
	File Reference	4083
7.454	src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h	
	File Reference	4084
7.455	src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h	
	File Reference	4084
7.456	src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h	
	File Reference	4085
7.457	src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h	
	File Reference	4086
7.458	src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h	
	File Reference	4086
7.459	src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h	
	File Reference	4087
7.460	src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h	
	File Reference	4088
7.461	src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h	
	File Reference	4088
7.462	src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h	
	File Reference	4089
7.463	src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h	
	File Reference	4090
7.464	src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h	
	File Reference	4090
7.465	src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h	
	File Reference	4091
7.466	src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarshaller.h	
	File Reference	4092
7.467	src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h	
	File Reference	4092
7.468	src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h	
	File Reference	4093

7.469	src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h	
	File Reference	4094
7.470	src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h	
	File Reference	4094
7.471	src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h	
	File Reference	4095
7.472	src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h	
	File Reference	4096
7.473	src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h	
	File Reference	4096
7.474	src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h	
	File Reference	4097
7.475	src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h	
	File Reference	4098
7.476	src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h	
	File Reference	4098
7.477	src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h	
	File Reference	4099
7.478	src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h	
	File Reference	4100
7.479	src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h	
	File Reference	4100
7.480	src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h	
	File Reference	4101
7.481	src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h	
	File Reference	4102
7.482	src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h	
	File Reference	4102
7.483	src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h	
	File Reference	4103
7.484	src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscriptionInfoMarshaller.h	
	File Reference	4104
7.485	src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h	
	File Reference	4104
7.486	src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h	
	File Reference	4105
7.487	src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h	
	File Reference	4106
7.488	src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h	
	File Reference	4106
7.489	src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h	
	File Reference	4107
7.490	src/main/activemq/wireformat/openwire/marshal/v6/ReplayCommandMarshaller.h	
	File Reference	4108

7.491	src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h	File	
	Reference		4108
7.492	src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h	File	
	Reference		4109
7.493	src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h	File	
	Reference		4110
7.494	src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h	File	
	Reference		4110
7.495	src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h	File	
	Reference		4111
7.496	src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h	File	
	Reference		4112
7.497	src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h	File	
	Reference		4112
7.498	src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h	File	
	Reference		4113
7.499	src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h	File	
	Reference		4114
7.500	src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h	File	
	Reference		4114
7.501	src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h	File	
	Reference		4115
7.502	src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h	File	
	Reference		4116
7.503	src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h	File	
	File Reference		4116
7.504	src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h	File	
	File Reference		4117
7.505	src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h	File	
	File Reference		4118
7.506	src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h	File	
	File Reference		4118
7.507	src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h	File	
	File Reference		4119
7.508	src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h	File	
	File Reference		4120
7.509	src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h	File	
	File Reference		4120
7.510	src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h	File	
	File Reference		4121
7.511	src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h	File	
	File Reference		4122
7.512	src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h	File	
	File Reference		4122

7.513src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h	
File Reference	4123
7.514src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h	
File Reference	4124
7.515src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h	
File Reference	4124
7.516src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h	
File Reference	4125
7.517src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h	
File Reference	4126
7.518src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h	
File Reference	4126
7.519src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h	
File Reference	4127
7.520src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h	
File Reference	4128
7.521src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h	
File Reference	4128
7.522src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h	
File Reference	4129
7.523src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h	
File Reference	4130
7.524src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h	
File Reference	4130
7.525src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h	
File Reference	4131
7.526src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h	
File Reference	4132
7.527src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h	
File Reference	4132
7.528src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h	
File Reference	4133
7.529src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h	
File Reference	4134
7.530src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h	
File Reference	4134
7.531src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h	
File Reference	4135
7.532src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h	
File Reference	4136
7.533src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h	
File Reference	4136
7.534src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h	
File Reference	4137

7.535	src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h	
	File Reference	4138
7.536	src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h	
	File Reference	4138
7.537	src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h	
	File Reference	4139
7.538	src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h	
	File Reference	4140
7.539	src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h	
	File Reference	4140
7.540	src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h	
	File Reference	4141
7.541	src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h	
	File Reference	4142
7.542	src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h	
	File Reference	4142
7.543	src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h	
	File Reference	4143
7.544	src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h	
	File Reference	4144
7.545	src/main/activemq/wireformat/openwire/OpenWireFormat.h	File Reference 4144
7.546	src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h	File Reference 4145
7.547	src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h	File Reference 4146
7.548	src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h	File Reference 4146
7.549	src/main/activemq/wireformat/openwire/utils/BooleanStream.h	File Reference 4147
7.550	src/main/activemq/wireformat/openwire/utils/HexTable.h	File Reference 4147
7.551	src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h	File Reference 4148
7.552	src/main/activemq/wireformat/stomp/StompCommandConstants.h	File Reference 4148
7.553	src/main/activemq/wireformat/stomp/StompFrame.h	File Reference 4149
7.554	src/main/activemq/wireformat/stomp/StompHelper.h	File Reference 4150
7.555	src/main/activemq/wireformat/stomp/StompWireFormat.h	File Reference 4150
7.556	src/main/activemq/wireformat/stomp/StompWireFormatFactory.h	File Reference 4151
7.557	src/main/activemq/wireformat/WireFormat.h	File Reference 4151
7.558	src/main/activemq/wireformat/WireFormatFactory.h	File Reference 4152
7.559	src/main/activemq/wireformat/WireFormatNegotiator.h	File Reference 4153
7.560	src/main/activemq/wireformat/WireFormatRegistry.h	File Reference 4153
7.561	src/main/cms/BytesMessage.h	File Reference 4154
7.562	src/main/cms/Closeable.h	File Reference 4154

7.563src/main/decaf/io/Closeable.h File Reference	4155
7.564src/main/cms/CMSException.h File Reference	4155
7.565src/main/cms/CMSProperties.h File Reference	4156
7.566src/main/cms/CMSSecurityException.h File Reference	4156
7.567src/main/cms/Connection.h File Reference	4156
7.568src/main/cms/ConnectionFactory.h File Reference	4157
7.569src/main/cms/ConnectionMetaData.h File Reference	4157
7.570src/main/cms/DeliveryMode.h File Reference	4158
7.571src/main/cms/Destination.h File Reference	4158
7.572src/main/cms/ExceptionListener.h File Reference	4159
7.573src/main/cms/IllegalStateException.h File Reference	4159
7.574src/main/decaf/lang/exceptions/IllegalStateException.h File Reference	4160
7.575src/main/cms/InvalidClientIdException.h File Reference	4160
7.576src/main/cms/InvalidDestinationException.h File Reference	4160
7.577src/main/cms/InvalidSelectorException.h File Reference	4161
7.578src/main/cms/MapMessage.h File Reference	4161
7.579src/main/cms/MessageConsumer.h File Reference	4162
7.580src/main/cms/MessageEnumeration.h File Reference	4162
7.581src/main/cms/MessageEOFException.h File Reference	4163
7.582src/main/cms/MessageFormatException.h File Reference	4163
7.583src/main/cms/MessageListener.h File Reference	4164
7.584src/main/cms/MessageNotReadableException.h File Reference	4164
7.585src/main/cms/MessageNotWriteableException.h File Reference	4164
7.586src/main/cms/MessageProducer.h File Reference	4165
7.587src/main/cms/ObjectMessage.h File Reference	4165
7.588src/main/cms/Queue.h File Reference	4166
7.589src/main/decaf/util/Queue.h File Reference	4166
7.590src/main/cms/QueueBrowser.h File Reference	4167
7.591src/main/cms/Session.h File Reference	4167
7.592src/main/cms/Startable.h File Reference	4168
7.593src/main/cms/Stoppable.h File Reference	4169
7.594src/main/cms/StreamMessage.h File Reference	4169
7.595src/main/cms/TemporaryQueue.h File Reference	4169
7.596src/main/cms/TemporaryTopic.h File Reference	4170
7.597src/main/cms/TextMessage.h File Reference	4170
7.598src/main/cms/Topic.h File Reference	4171

7.599src/main/cms/UnsupportedOperationException.h File Reference	4171
7.600src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference	4172
7.601src/main/decaf/internal/AprPool.h File Reference	4172
7.602src/main/decaf/internal/DecafRuntime.h File Reference	4173
7.603src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference	4173
7.604src/main/decaf/internal/io/StandardInputStream.h File Reference	4174
7.605src/main/decaf/internal/io/StandardOutputStream.h File Reference	4174
7.606src/main/decaf/internal/net/DefaultServerSocketFactory.h File Reference	4174
7.607src/main/decaf/internal/net/DefaultSocketFactory.h File Reference	4175
7.608src/main/decaf/internal/net/Network.h File Reference	4175
7.609src/main/decaf/internal/net/SocketFileDescriptor.h File Reference	4176
7.610src/main/decaf/internal/net/ssl/DefaultSSLContext.h File Reference	4176
7.611src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h File Reference	4177
7.612src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Reference	4177
7.613src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h File Reference	4178
7.614src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h File Reference	4179
7.615src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h File Reference	4179
7.616src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h File Reference	4180
7.617src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h File Reference	4180
7.618src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h File Reference	4181
7.619src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h File Reference	4181
7.620src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h File Reference	4182
7.621src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h File Reference	4182
7.622src/main/decaf/internal/net/tcp/TcpSocket.h File Reference	4183
7.623src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference	4184
7.624src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h File Reference	4184
7.625src/main/decaf/internal/net/URIEncoderDecoder.h File Reference	4185
7.626src/main/decaf/internal/net/URIHelper.h File Reference	4185
7.627src/main/decaf/internal/net/URIType.h File Reference	4186
7.628src/main/decaf/internal/nio/BufferFactory.h File Reference	4186
7.629src/main/decaf/internal/nio/ByteArrayBuffer.h File Reference	4187
7.630src/main/decaf/internal/nio/CharArrayBuffer.h File Reference	4188
7.631src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference	4188
7.632src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference	4189

7.633src/main/decaf/internal/nio/IntArrayBuffer.h File Reference	4189
7.634src/main/decaf/internal/nio/LongArrayBuffer.h File Reference	4190
7.635src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference	4191
7.636src/main/decaf/internal/security/unix/SecureRandomImpl.h File Reference	4191
7.637src/main/decaf/internal/security/windows/SecureRandomImpl.h File Reference . .	4192
7.638src/main/decaf/internal/util/ByteArrayAdapter.h File Reference	4192
7.639src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference	4193
7.640src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference	4193
7.641src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference . . .	4194
7.642src/main/decaf/internal/util/concurrent/Transferer.h File Reference	4194
7.643src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference	4195
7.644src/main/decaf/internal/util/concurrent/TransferStack.h File Reference	4195
7.645src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference .	4196
7.646src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference	4196
7.647src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference . . .	4197
7.648src/main/decaf/internal/util/concurrent/windows/MutexHandle.h File Reference .	4197
7.649src/main/decaf/internal/util/GenericResource.h File Reference	4197
7.650src/main/decaf/internal/util/HexStringParser.h File Reference	4198
7.651src/main/decaf/internal/util/Resource.h File Reference	4198
7.652src/main/decaf/internal/util/TimerTaskHeap.h File Reference	4199
7.653src/main/decaf/internal/util/zip/crc32.h File Reference	4199
7.653.1 Variable Documentation	4199
7.653.1.1 crc_table	4199
7.654src/main/decaf/internal/util/zip/deflate.h File Reference	4199
7.654.1 Define Documentation	4201
7.654.1.1 _tr_tally_dist	4201
7.654.1.2 _tr_tally_lit	4201
7.654.1.3 BL_CODES	4202
7.654.1.4 BUSY_STATE	4202
7.654.1.5 Code	4202
7.654.1.6 COMMENT_STATE	4202
7.654.1.7 d_code	4202
7.654.1.8 D_CODES	4202
7.654.1.9 Dad	4202
7.654.1.10EXTRA_STATE	4202
7.654.1.11FINISH_STATE	4202

7.654.1.12	Freq	4202
7.654.1.13	GZIP	4202
7.654.1.14	HCRC_STATE	4202
7.654.1.15	HEAP_SIZE	4202
7.654.1.16	INIT_STATE	4202
7.654.1.17	L_CODES	4202
7.654.1.18	Len	4202
7.654.1.19	LENGTH_CODES	4202
7.654.1.20	LITERALS	4202
7.654.1.21	MAX_BITS	4202
7.654.1.22	MAX_DIST	4202
7.654.1.23	max_insert_length	4202
7.654.1.24	MIN_LOOKAHEAD	4202
7.654.1.25	NAME_STATE	4202
7.654.1.26	put_byte	4202
7.654.1.27	WIN_INIT	4202
7.654.2	Typedef Documentation	4202
7.654.2.1	ct_data	4202
7.654.2.2	deflate_state	4202
7.654.2.3	IPos	4202
7.654.2.4	Pos	4202
7.654.2.5	Posf	4202
7.654.2.6	static_tree_desc	4202
7.654.2.7	tree_desc	4202
7.654.3	Function Documentation	4202
7.654.3.1	OF	4202
7.654.3.2	OF	4202
7.654.3.3	OF	4202
7.654.4	Variable Documentation	4202
7.654.4.1	_dist_code	4202
7.654.4.2	_length_code	4202
7.655	src/main/decaf/internal/util/zip/gzguts.h File Reference	4202
7.655.1	Define Documentation	4204
7.655.1.1	COPY	4204
7.655.1.2	GT_OFF	4204
7.655.1.3	GZ_APPEND	4204

7.655.1.4 GZ_NONE	4204
7.655.1.5 GZ_READ	4204
7.655.1.6 GZ_WRITE	4204
7.655.1.7 GZBUFSIZE	4204
7.655.1.8 GZIP	4204
7.655.1.9 local	4204
7.655.1.10 LOOK	4204
7.655.1.11 ZLIB_INTERNAL	4204
7.655.1.12 zstrerror	4204
7.655.2 Typedef Documentation	4204
7.655.2.1 gz_statep	4204
7.655.3 Function Documentation	4204
7.655.3.1 OF	4204
7.655.3.2 OF	4204
7.655.3.3 OF	4204
7.655.3.4 OF	4204
7.655.3.5 OF	4204
7.655.3.6 OF	4204
7.655.3.7 OF	4204
7.656src/main/decaf/internal/util/zip/inffast.h File Reference	4204
7.656.1 Function Documentation	4205
7.656.1.1 OF	4205
7.657src/main/decaf/internal/util/zip/inffixed.h File Reference	4205
7.658src/main/decaf/internal/util/zip/inflate.h File Reference	4205
7.658.1 Define Documentation	4205
7.658.1.1 GUNZIP	4205
7.658.2 Enumeration Type Documentation	4205
7.658.2.1 inflate_mode	4205
7.659src/main/decaf/internal/util/zip/inftrees.h File Reference	4206
7.659.1 Define Documentation	4207
7.659.1.1 ENOUGH	4207
7.659.1.2 ENOUGH_DISTS	4207
7.659.1.3 ENOUGH_LENS	4207
7.659.2 Enumeration Type Documentation	4207
7.659.2.1 codetype	4207
7.659.3 Function Documentation	4207

7.659.3.1 OF	4207
7.660src/main/decaf/internal/util/zip/trees.h File Reference	4207
7.660.1 Variable Documentation	4207
7.660.1.1 _dist_code	4207
7.660.1.2 _length_code	4208
7.660.1.3 base_dist	4208
7.660.1.4 base_length	4209
7.660.1.5 static_dtree	4209
7.660.1.6 static_ltree	4209
7.661src/main/decaf/internal/util/zip/zconf.h File Reference	4209
7.661.1 Define Documentation	4211
7.661.1.1 const	4211
7.661.1.2 FAR	4211
7.661.1.3 MAX_MEM_LEVEL	4211
7.661.1.4 MAX_WBITS	4211
7.661.1.5 OF	4211
7.661.1.6 SEEK_CUR	4211
7.661.1.7 SEEK_END	4211
7.661.1.8 SEEK_SET	4211
7.661.1.9 z_off64_t	4211
7.661.1.10z_off_t	4211
7.661.1.11ZEXPORT	4211
7.661.1.12ZEXPORTVA	4211
7.661.1.13ZEXTERN	4211
7.661.2 Typedef Documentation	4211
7.661.2.1 Byte	4211
7.661.2.2 Bytef	4211
7.661.2.3 charf	4211
7.661.2.4 intf	4211
7.661.2.5 uInt	4211
7.661.2.6 uIntf	4211
7.661.2.7 uLong	4211
7.661.2.8 uLongf	4211
7.661.2.9 voidp	4211
7.661.2.10voidpc	4211
7.661.2.11voidpf	4211

7.662src/main/decaf/internal/util/zip/zlib.h File Reference	4211
7.662.1 Define Documentation	4214
7.662.1.1 deflateInit	4214
7.662.1.2 deflateInit2	4214
7.662.1.3 inflateBackInit	4214
7.662.1.4 inflateInit	4215
7.662.1.5 inflateInit2	4215
7.662.1.6 Z_ASCII	4215
7.662.1.7 Z_BEST_COMPRESSION	4215
7.662.1.8 Z_BEST_SPEED	4215
7.662.1.9 Z_BINARY	4215
7.662.1.10Z_BLOCK	4215
7.662.1.11Z_BUF_ERROR	4215
7.662.1.12Z_DATA_ERROR	4215
7.662.1.13Z_DEFAULT_COMPRESSION	4215
7.662.1.14Z_DEFAULT_STRATEGY	4215
7.662.1.15Z_DEFLATED	4215
7.662.1.16Z_ERRNO	4215
7.662.1.17Z_FILTERED	4215
7.662.1.18Z_FINISH	4215
7.662.1.19Z_FIXED	4215
7.662.1.20Z_FULL_FLUSH	4215
7.662.1.21Z_HUFFMAN_ONLY	4215
7.662.1.22Z_MEM_ERROR	4215
7.662.1.23Z_NEED_DICT	4215
7.662.1.24Z_NO_COMPRESSION	4215
7.662.1.25Z_NO_FLUSH	4215
7.662.1.26Z_NULL	4215
7.662.1.27Z_OK	4215
7.662.1.28Z_PARTIAL_FLUSH	4215
7.662.1.29Z_RLE	4215
7.662.1.30Z_STREAM_END	4215
7.662.1.31Z_STREAM_ERROR	4215
7.662.1.32Z_SYNC_FLUSH	4215
7.662.1.33Z_TEXT	4215
7.662.1.34Z_TREES	4215

7.662.1.35	Z_UNKNOWN	4215
7.662.1.36	Z_VERSION_ERROR	4215
7.662.1.37	ZLIB_VER_MAJOR	4215
7.662.1.38	ZLIB_VER_MINOR	4215
7.662.1.39	ZLIB_VER_REVISION	4215
7.662.1.40	ZLIB_VER_SUBREVISION	4215
7.662.1.41	ZLIB_VERNUM	4215
7.662.1.42	lib_version	4215
7.662.1.43	ZLIB_VERSION	4215
7.662.2	Typedef Documentation	4215
7.662.2.1	gz_header	4215
7.662.2.2	gz_headerp	4215
7.662.2.3	gzFile	4215
7.662.2.4	OF	4215
7.662.2.5	z_stream	4215
7.662.2.6	z_streamp	4215
7.662.3	Function Documentation	4215
7.662.3.1	OF	4215
7.662.3.2	OF	4215
7.662.3.3	OF	4215
7.662.3.4	OF	4215
7.662.3.5	OF	4215
7.662.3.6	OF	4215
7.662.3.7	OF	4215
7.662.3.8	OF	4215
7.662.3.9	OF	4215
7.662.3.10	OF	4215
7.662.3.11	OF	4215
7.662.3.12	OF	4215
7.662.3.13	OF	4215
7.662.3.14	OF	4215
7.662.3.15	OF	4215
7.662.3.16	OF	4215
7.662.3.17	OF	4215
7.662.3.18	OF	4215
7.662.3.19	OF	4215

7.662.3.20OF	4215
7.662.3.21OF	4215
7.662.3.22OF	4215
7.662.3.23OF	4215
7.662.3.24OF	4215
7.662.3.25OF	4215
7.662.3.26OF	4215
7.662.3.27OF	4215
7.662.3.28OF	4215
7.662.3.29OF	4215
7.662.3.30OF	4215
7.662.3.31OF	4215
7.662.3.32OF	4215
7.662.3.33OF	4215
7.662.3.34OF	4215
7.662.3.35OF	4215
7.662.3.36OF	4215
7.662.3.37OF	4215
7.662.3.38OF	4215
7.662.3.39OF	4215
7.662.3.40OF	4215
7.662.3.41OF	4215
7.662.3.42OF	4215
7.663src/main/decaf/internal/util/zip/zutil.h File Reference	4215
7.663.1 Define Documentation	4217
7.663.1.1 Assert	4217
7.663.1.2 DEF_MEM_LEVEL	4217
7.663.1.3 DEF_WBITS	4217
7.663.1.4 DYN_TREES	4217
7.663.1.5 ERR_MSG	4217
7.663.1.6 ERR_RETURN	4217
7.663.1.7 F_OPEN	4217
7.663.1.8 local	4217
7.663.1.9 MAX_MATCH	4217
7.663.1.10MIN_MATCH	4217
7.663.1.11OS_CODE	4217

7.663.1.12	PRESET_DICT	4217
7.663.1.13	STATIC_TREES	4217
7.663.1.14	STORED_BLOCK	4217
7.663.1.15	Trace	4217
7.663.1.16	Tracec	4217
7.663.1.17	Tracecv	4217
7.663.1.18	Tracev	4217
7.663.1.19	Tracevv	4217
7.663.1.20	TRY_FREE	4217
7.663.1.21	ZALLOC	4217
7.663.1.22	ZFREE	4217
7.663.1.23	ZLIB_INTERNAL	4217
7.663.2	Typedef Documentation	4217
7.663.2.1	uch	4217
7.663.2.2	uchf	4217
7.663.2.3	ulg	4217
7.663.2.4	ush	4217
7.663.2.5	ushf	4217
7.663.3	Function Documentation	4217
7.663.3.1	OF	4217
7.663.3.2	OF	4217
7.663.3.3	OF	4217
7.663.3.4	OF	4217
7.663.3.5	OF	4217
7.663.3.6	OF	4217
7.663.4	Variable Documentation	4217
7.663.4.1	z_errmsg	4217
7.664	src/main/decaf/io/BlockingByteArrayInputStream.h File Reference	4217
7.665	src/main/decaf/io/BufferedInputStream.h File Reference	4218
7.666	src/main/decaf/io/BufferedOutputStream.h File Reference	4218
7.667	src/main/decaf/io/ByteArrayInputStream.h File Reference	4219
7.668	src/main/decaf/io/ByteArrayOutputStream.h File Reference	4219
7.669	src/main/decaf/io/DataInput.h File Reference	4220
7.670	src/main/decaf/io/DataInputStream.h File Reference	4220
7.671	src/main/decaf/io/DataOutput.h File Reference	4221
7.672	src/main/decaf/io/DataOutputStream.h File Reference	4222

7.673src/main/decaf/io/EOFException.h File Reference	4222
7.674src/main/decaf/io/FileDescriptor.h File Reference	4222
7.675src/main/decaf/io/FilterInputStream.h File Reference	4223
7.676src/main/decaf/io/FilterOutputStream.h File Reference	4223
7.677src/main/decaf/io/Flushable.h File Reference	4224
7.678src/main/decaf/io/InputStream.h File Reference	4224
7.679src/main/decaf/io/InputStreamReader.h File Reference	4225
7.680src/main/decaf/io/InterruptedIOException.h File Reference	4225
7.681src/main/decaf/io/IOException.h File Reference	4226
7.682src/main/decaf/io/OutputStream.h File Reference	4226
7.683src/main/decaf/io/OutputStreamWriter.h File Reference	4227
7.684src/main/decaf/io/PushbackInputStream.h File Reference	4227
7.685src/main/decaf/io/Reader.h File Reference	4228
7.686src/main/decaf/io/UnsupportedEncodingException.h File Reference	4228
7.687src/main/decaf/io/UTFDataFormatException.h File Reference	4229
7.688src/main/decaf/io/Writer.h File Reference	4229
7.689src/main/decaf/lang/Appendable.h File Reference	4230
7.690src/main/decaf/lang/ArrayPointer.h File Reference	4230
7.691src/main/decaf/lang/Boolean.h File Reference	4231
7.692src/main/decaf/lang/Byte.h File Reference	4232
7.693src/main/decaf/lang/Character.h File Reference	4232
7.694src/main/decaf/lang/CharSequence.h File Reference	4232
7.695src/main/decaf/lang/Comparable.h File Reference	4233
7.696src/main/decaf/lang/Double.h File Reference	4233
7.697src/main/decaf/lang/Exception.h File Reference	4234
7.698src/main/decaf/lang/exceptions/ClassCastException.h File Reference	4234
7.699src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference	4235
7.700src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference	4235
7.701src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference	4235
7.702src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference	4236
7.703src/main/decaf/lang/exceptions/InterruptedException.h File Reference	4236
7.704src/main/decaf/lang/exceptions/InvalidStateException.h File Reference	4237
7.705src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference	4237
7.706src/main/decaf/lang/exceptions/NullPointerException.h File Reference	4237
7.707src/main/decaf/lang/exceptions/NumberFormatException.h File Reference	4238
7.708src/main/decaf/lang/exceptions/RuntimeException.h File Reference	4238

7.709src/main/decaf/lang/Float.h File Reference	4239
7.710src/main/decaf/lang/Integer.h File Reference	4239
7.711src/main/decaf/lang/Iterable.h File Reference	4239
7.712src/main/decaf/lang/Long.h File Reference	4240
7.713src/main/decaf/lang/Math.h File Reference	4240
7.714src/main/decaf/lang/Number.h File Reference	4241
7.715src/main/decaf/lang/Pointer.h File Reference	4241
7.716src/main/decaf/lang/Readable.h File Reference	4242
7.717src/main/decaf/lang/Runnable.h File Reference	4243
7.718src/main/decaf/lang/Runtime.h File Reference	4243
7.719src/main/decaf/lang/Short.h File Reference	4244
7.720src/main/decaf/lang/String.h File Reference	4244
7.721src/main/decaf/lang/System.h File Reference	4244
7.722src/main/decaf/lang/Thread.h File Reference	4245
7.723src/main/decaf/lang/ThreadGroup.h File Reference	4246
7.724src/main/decaf/lang/Throwable.h File Reference	4246
7.725src/main/decaf/net/BindException.h File Reference	4247
7.726src/main/decaf/net/ConnectException.h File Reference	4247
7.727src/main/decaf/net/HttpRetryException.h File Reference	4247
7.728src/main/decaf/net/Inet4Address.h File Reference	4248
7.729src/main/decaf/net/Inet6Address.h File Reference	4248
7.730src/main/decaf/net/InetAddress.h File Reference	4249
7.731src/main/decaf/net/InetSocketAddress.h File Reference	4249
7.732src/main/decaf/net/MalformedURLException.h File Reference	4249
7.733src/main/decaf/net/NoRouteToHostException.h File Reference	4250
7.734src/main/decaf/net/PortUnreachableException.h File Reference	4250
7.735src/main/decaf/net/ProtocolException.h File Reference	4251
7.736src/main/decaf/net/ServerSocket.h File Reference	4251
7.737src/main/decaf/net/ServerSocketFactory.h File Reference	4252
7.738src/main/decaf/net/Socket.h File Reference	4252
7.739src/main/decaf/net/SocketAddress.h File Reference	4253
7.740src/main/decaf/net/SocketError.h File Reference	4253
7.741src/main/decaf/net/SocketException.h File Reference	4254
7.742src/main/decaf/net/SocketFactory.h File Reference	4254
7.743src/main/decaf/net/SocketImpl.h File Reference	4255
7.744src/main/decaf/net/SocketImplFactory.h File Reference	4255

7.745	src/main/decaf/net/SocketOptions.h File Reference	4256
7.746	src/main/decaf/net/SocketTimeoutException.h File Reference	4256
7.747	src/main/decaf/net/ssl/SSLContext.h File Reference	4256
7.748	src/main/decaf/net/ssl/SSLContextSpi.h File Reference	4257
7.749	src/main/decaf/net/ssl/SSLParameters.h File Reference	4257
7.750	src/main/decaf/net/ssl/SSLServerSocket.h File Reference	4258
7.751	src/main/decaf/net/ssl/SSLServerSocketFactory.h File Reference	4258
7.752	src/main/decaf/net/ssl/SSLSocket.h File Reference	4259
7.753	src/main/decaf/net/ssl/SSLSocketFactory.h File Reference	4259
7.754	src/main/decaf/net/UnknownHostException.h File Reference	4260
7.755	src/main/decaf/net/UnknownServiceException.h File Reference	4260
7.756	src/main/decaf/net/URI.h File Reference	4261
7.757	src/main/decaf/net/URISyntaxException.h File Reference	4261
7.758	src/main/decaf/net/URL.h File Reference	4262
7.759	src/main/decaf/net/URLDecoder.h File Reference	4262
7.760	src/main/decaf/net/URLEncoder.h File Reference	4262
7.761	src/main/decaf/nio/Buffer.h File Reference	4263
7.762	src/main/decaf/nio/BufferOverflowException.h File Reference	4263
7.763	src/main/decaf/nio/BufferUnderflowException.h File Reference	4264
7.764	src/main/decaf/nio/ByteBuffer.h File Reference	4264
7.765	src/main/decaf/nio/CharBuffer.h File Reference	4265
7.766	src/main/decaf/nio/DoubleBuffer.h File Reference	4265
7.767	src/main/decaf/nio/FloatBuffer.h File Reference	4266
7.768	src/main/decaf/nio/IntBuffer.h File Reference	4266
7.769	src/main/decaf/nio/InvalidMarkException.h File Reference	4267
7.770	src/main/decaf/nio/LongBuffer.h File Reference	4267
7.771	src/main/decaf/nio/ReadOnlyBufferException.h File Reference	4268
7.772	src/main/decaf/nio/ShortBuffer.h File Reference	4268
7.773	src/main/decaf/security/auth/x500/X500Principal.h File Reference	4269
7.774	src/main/decaf/security/cert/Certificate.h File Reference	4269
7.775	src/main/decaf/security/cert/CertificateEncodingException.h File Reference . . .	4270
7.776	src/main/decaf/security/cert/CertificateException.h File Reference	4270
7.777	src/main/decaf/security/cert/CertificateExpiredException.h File Reference	4271
7.778	src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference . .	4271
7.779	src/main/decaf/security/cert/CertificateParsingException.h File Reference	4272
7.780	src/main/decaf/security/cert/X509Certificate.h File Reference	4272

7.781src/main/decaf/security/GeneralSecurityException.h File Reference	4273
7.782src/main/decaf/security/InvalidKeyException.h File Reference	4273
7.783src/main/decaf/security/Key.h File Reference	4273
7.784src/main/decaf/security/KeyException.h File Reference	4274
7.785src/main/decaf/security/KeyManagementException.h File Reference	4274
7.786src/main/decaf/security/NoSuchAlgorithmException.h File Reference	4275
7.787src/main/decaf/security/NoSuchProviderException.h File Reference	4275
7.788src/main/decaf/security/Principal.h File Reference	4275
7.789src/main/decaf/security/PublicKey.h File Reference	4276
7.790src/main/decaf/security/SecureRandom.h File Reference	4276
7.791src/main/decaf/security/SecureRandomSpi.h File Reference	4277
7.792src/main/decaf/security/SignatureException.h File Reference	4277
7.793src/main/decaf/util/AbstractCollection.h File Reference	4278
7.794src/main/decaf/util/AbstractList.h File Reference	4278
7.795src/main/decaf/util/AbstractMap.h File Reference	4279
7.796src/main/decaf/util/AbstractQueue.h File Reference	4279
7.797src/main/decaf/util/AbstractSequentialList.h File Reference	4280
7.798src/main/decaf/util/AbstractSet.h File Reference	4281
7.799src/main/decaf/util/Collection.h File Reference	4281
7.800src/main/decaf/util/Comparator.h File Reference	4282
7.801src/main/decaf/util/comparators/Less.h File Reference	4282
7.802src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference	4283
7.803src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference	4283
7.804src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h File Reference	4284
7.805src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference	4284
7.806src/main/decaf/util/concurrent/BlockingQueue.h File Reference	4285
7.807src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference	4285
7.808src/main/decaf/util/concurrent/Callable.h File Reference	4286
7.809src/main/decaf/util/concurrent/CancellationException.h File Reference	4286
7.810src/main/decaf/util/concurrent/Concurrent.h File Reference	4287
7.810.1 Define Documentation	4287
7.810.1.1 synchronized	4287
7.810.1.2 WAIT_INFINITE	4287
7.811src/main/decaf/util/concurrent/ConcurrentMap.h File Reference	4288
7.812src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference	4288
7.813src/main/decaf/util/concurrent/CountDownLatch.h File Reference	4289

7.814src/main/decaf/util/concurrent/Delayed.h File Reference	4289
7.815src/main/decaf/util/concurrent/ExecutionException.h File Reference	4290
7.816src/main/decaf/util/concurrent/Executor.h File Reference	4290
7.817src/main/decaf/util/concurrent/ExecutorService.h File Reference	4291
7.818src/main/decaf/util/concurrent/Future.h File Reference	4291
7.819src/main/decaf/util/concurrent/Lock.h File Reference	4292
7.820src/main/decaf/util/concurrent/locks/Lock.h File Reference	4292
7.821src/main/decaf/util/concurrent/locks/Condition.h File Reference	4293
7.822src/main/decaf/util/concurrent/locks/LockSupport.h File Reference	4293
7.823src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference	4294
7.824src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference	4294
7.825src/main/decaf/util/concurrent/Mutex.h File Reference	4295
7.826src/main/decaf/util/concurrent/PooledThread.h File Reference	4295
7.827src/main/decaf/util/concurrent/PooledThreadListener.h File Reference	4296
7.828src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference . . .	4296
7.829src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference . . .	4296
7.830src/main/decaf/util/concurrent/Semaphore.h File Reference	4297
7.831src/main/decaf/util/concurrent/Synchronizable.h File Reference	4297
7.832src/main/decaf/util/concurrent/SynchronousQueue.h File Reference	4298
7.833src/main/decaf/util/concurrent/TaskListener.h File Reference	4299
7.834src/main/decaf/util/concurrent/ThreadFactory.h File Reference	4299
7.835src/main/decaf/util/concurrent/ThreadPool.h File Reference	4299
7.836src/main/decaf/util/concurrent/TimeoutException.h File Reference	4300
7.837src/main/decaf/util/concurrent/TimeUnit.h File Reference	4301
7.838src/main/decaf/util/Date.h File Reference	4301
7.839src/main/decaf/util/Iterator.h File Reference	4302
7.840src/main/decaf/util/List.h File Reference	4302
7.841src/main/decaf/util/ListIterator.h File Reference	4303
7.842src/main/decaf/util/logging/ConsoleHandler.h File Reference	4303
7.843src/main/decaf/util/logging/ErrorHandler.h File Reference	4304
7.844src/main/decaf/util/logging/Filter.h File Reference	4304
7.845src/main/decaf/util/logging/Formatter.h File Reference	4305
7.846src/main/decaf/util/logging/Handler.h File Reference	4305
7.847src/main/decaf/util/logging/Level.h File Reference	4306
7.848src/main/decaf/util/logging/Logger.h File Reference	4306
7.849src/main/decaf/util/logging/LoggerCommon.h File Reference	4307

7.850src/main/decaf/util/logging/LoggerDefines.h File Reference	4307
7.850.1 Define Documentation	4308
7.850.1.1 LOGDECAF_DEBUG	4308
7.850.1.2 LOGDECAF_DEBUG_1	4308
7.850.1.3 LOGDECAF_DECLARE	4309
7.850.1.4 LOGDECAF_DECLARE_LOCAL	4309
7.850.1.5 LOGDECAF_ERROR	4309
7.850.1.6 LOGDECAF_FATAL	4309
7.850.1.7 LOGDECAF_INFO	4309
7.850.1.8 LOGDECAF_INITIALIZE	4309
7.850.1.9 LOGDECAF_WARN	4309
7.851src/main/decaf/util/logging/LoggerHierarchy.h File Reference	4309
7.852src/main/decaf/util/logging/LogManager.h File Reference	4309
7.853src/main/decaf/util/logging/LogRecord.h File Reference	4310
7.854src/main/decaf/util/logging/LogWriter.h File Reference	4311
7.855src/main/decaf/util/logging/MarkBlockLogger.h File Reference	4311
7.856src/main/decaf/util/logging/PropertiesChangeListener.h File Reference	4312
7.857src/main/decaf/util/logging/SimpleFormatter.h File Reference	4312
7.858src/main/decaf/util/logging/SimpleLogger.h File Reference	4313
7.859src/main/decaf/util/logging/StreamHandler.h File Reference	4313
7.860src/main/decaf/util/logging/XMLFormatter.h File Reference	4314
7.861src/main/decaf/util/Map.h File Reference	4314
7.862src/main/decaf/util/PriorityQueue.h File Reference	4315
7.863src/main/decaf/util/Properties.h File Reference	4315
7.864src/main/decaf/util/Random.h File Reference	4316
7.865src/main/decaf/util/Set.h File Reference	4316
7.866src/main/decaf/util/StlList.h File Reference	4317
7.867src/main/decaf/util/StlMap.h File Reference	4318
7.868src/main/decaf/util/StlQueue.h File Reference	4318
7.869src/main/decaf/util/StlSet.h File Reference	4319
7.870src/main/decaf/util/StringTokenizer.h File Reference	4319
7.871src/main/decaf/util/Timer.h File Reference	4320
7.872src/main/decaf/util/TimerTask.h File Reference	4320
7.873src/main/decaf/util/UUID.h File Reference	4321
7.874src/main/decaf/util/zip/Adler32.h File Reference	4321
7.875src/main/decaf/util/zip/CheckedInputStream.h File Reference	4322

7.876	src/main/decaf/util/zip/CheckedOutputStream.h File Reference	4322
7.877	src/main/decaf/util/zip/Checksum.h File Reference	4323
7.878	src/main/decaf/util/zip/CRC32.h File Reference	4323
7.879	src/main/decaf/util/zip/DataFormatException.h File Reference	4324
7.880	src/main/decaf/util/zip/Deflater.h File Reference	4324
7.881	src/main/decaf/util/zip/DeflaterOutputStream.h File Reference	4325
7.882	src/main/decaf/util/zip/Inflater.h File Reference	4325
7.883	src/main/decaf/util/zip/InflaterInputStream.h File Reference	4326
7.884	src/main/decaf/util/zip/ZipException.h File Reference	4326

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

activemq (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements)	83
activemq::cmsutil	84
activemq::commands	85
activemq::core	86
activemq::core::policies	87
activemq::exceptions	87
activemq::io	87
activemq::library	88
activemq::state	88
activemq::threads	88
activemq::transport	89
activemq::transport::correlator	89
activemq::transport::failover	90
activemq::transport::inactivity	90
activemq::transport::logging	90
activemq::transport::mock	90
activemq::transport::tcp	91
activemq::util	91
activemq::wireformat	92
activemq::wireformat::openwire	92
activemq::wireformat::openwire::marshal	93
activemq::wireformat::openwire::marshal::v1	93
activemq::wireformat::openwire::marshal::v2	97
activemq::wireformat::openwire::marshal::v3	101
activemq::wireformat::openwire::marshal::v4	105
activemq::wireformat::openwire::marshal::v5	109
activemq::wireformat::openwire::marshal::v6	113
activemq::wireformat::openwire::utils	117
activemq::wireformat::stomp	117
cms (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements)	117

decaf (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements)	120
decaf::internal	121
decaf::internal::io	121
decaf::internal::net	122
decaf::internal::net::ssl	122
decaf::internal::net::ssl::openssl	123
decaf::internal::net::tcp	123
decaf::internal::nio	124
decaf::internal::security	124
decaf::internal::util	124
decaf::internal::util::concurrent	125
decaf::io	125
decaf::lang	127
decaf::lang::exceptions	130
decaf::net	130
decaf::net::ssl	132
decaf::nio	132
decaf::security	133
decaf::security::auth	133
decaf::security::auth::x500	133
decaf::security::cert	134
decaf::util	134
decaf::util::comparators	136
decaf::util::concurrent	136
decaf::util::concurrent::atomic	138
decaf::util::concurrent::locks	138
decaf::util::logging	139
decaf::util::zip	140
std	141

Chapter 2

Data Structure Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

activemq::core::ActiveMQAckHandler	165
activemq::core::ActiveMQConstants	265
activemq::library::ActiveMQCPP	277
activemq::core::ActiveMQTransactionContext	660
decaf::lang::Appendable	666
decaf::io::Writer	3756
decaf::io::OutputStreamWriter	2726
decaf::nio::CharBuffer	1037
decaf::internal::nio::CharArrayBuffer	1027
decaf::internal::AprPool	668
decaf::lang::ArrayPointer< T, REFCOUNTER >	669
decaf::util::concurrent::atomic::AtomicBoolean	677
decaf::util::concurrent::atomic::AtomicRefCounter	685
decaf::util::concurrent::atomic::AtomicReference< T >	687
binary_function	768
decaf::util::Comparator< ArrayPointer< T, R > >	1127
decaf::lang::ArrayPointerComparator< T, R >	676
decaf::util::Comparator< E >	1127
decaf::util::comparators::Less< E >	2182
decaf::util::Comparator< Pointer< T, R > >	1127
decaf::lang::PointerComparator< T, R >	2764
std::less< decaf::lang::ArrayPointer< T > >	2184
std::less< decaf::lang::Pointer< T > >	2185
activemq::wireformat::openwire::utils::BooleanStream	787
decaf::nio::Buffer	855
decaf::nio::ByteBuffer	954
decaf::internal::nio::ByteBuffer	915
decaf::nio::CharBuffer	1037
decaf::nio::DoubleBuffer	1692
decaf::internal::nio::DoubleArrayBuffer	1683
decaf::nio::FloatBuffer	1800
decaf::internal::nio::FloatArrayBuffer	1791

decaf::nio::IntBuffer	1931
decaf::internal::nio::IntArrayBuffer	1921
decaf::nio::LongBuffer	2291
decaf::internal::nio::LongArrayBuffer	2281
decaf::nio::ShortBuffer	3238
decaf::internal::nio::ShortArrayBuffer	3229
decaf::internal::nio::BufferFactory	869
decaf::internal::util::ByteArrayAdapter	893
decaf::util::concurrent::Callable< V >	1003
decaf::security::cert::Certificate	1007
decaf::security::cert::X509Certificate	3762
decaf::lang::CharSequence	1053
decaf::lang::String	3427
decaf::nio::CharBuffer	1037
decaf::util::zip::Checksum	1059
decaf::util::zip::Adler32	663
decaf::util::zip::CRC32	1420
cms::Closeable	1064
activemq::commands::ActiveMQTempDestination	523
activemq::commands::ActiveMQTempQueue	549
activemq::commands::ActiveMQTempTopic	576
cms::Connection	1168
activemq::core::ActiveMQConnection	233
cms::MessageConsumer	2423
activemq::cmsutil::CachedConsumer	993
activemq::core::ActiveMQConsumer	268
cms::MessageProducer	2550
activemq::cmsutil::CachedProducer	996
activemq::core::ActiveMQProducer	423
cms::QueueBrowser	2951
activemq::core::ActiveMQQueueBrowser	438
cms::Session	3148
activemq::cmsutil::PooledSession	2765
activemq::core::ActiveMQSession	465
decaf::io::Closeable	1065
activemq::transport::Transport	3629
activemq::transport::CompositeTransport	1135
activemq::transport::failover::FailoverTransport	1752
activemq::transport::IOTransport	2005
activemq::transport::mock::MockTransport	2592
activemq::transport::TransportFilter	3636
activemq::transport::correlator::ResponseCorrelator	3081
activemq::transport::inactivity::InactivityMonitor	1874
activemq::transport::logging::LoggingTransport	2252
activemq::transport::tcp::TcpTransport	3510
activemq::transport::tcp::SslTransport	3347
activemq::wireformat::WireFormatNegotiator	3751
activemq::wireformat::openwire::OpenWireFormatNegotiator	2713
decaf::io::InputStream	1909
decaf::internal::io::StandardInputStream	3352

decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream	2696
decaf::internal::net::tcp::TcpSocketInputStream	3506
decaf::io::BlockingByteArrayInputStream	771
decaf::io::ByteArrayInputStream	944
decaf::io::FilterInputStream	1771
activemq::io::LoggingInputStream	2250
decaf::io::BufferedInputStream	861
decaf::io::DataInputStream	1460
decaf::io::PushbackInputStream	2940
decaf::util::zip::CheckedInputStream	1055
decaf::util::zip::InflaterInputStream	1902
decaf::io::OutputStream	2718
decaf::internal::io::StandardErrorOutputStream	3351
decaf::internal::io::StandardOutputStream	3354
decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream	2698
decaf::internal::net::tcp::TcpSocketOutputStream	3509
decaf::io::ByteArrayOutputStream	951
decaf::io::FilterOutputStream	1777
activemq::io::LoggingOutputStream	2251
decaf::io::BufferedOutputStream	867
decaf::io::DataOutputStream	1473
decaf::util::zip::CheckedOutputStream	1058
decaf::util::zip::DeflaterOutputStream	1604
decaf::io::Reader	2960
decaf::io::InputStreamReader	1919
decaf::io::Writer	3756
decaf::net::Socket	3281
decaf::net::ssl::SSLSocket	3337
decaf::internal::net::ssl::openssl::OpenSSLSocket	2673
decaf::util::logging::Handler	1852
decaf::util::logging::StreamHandler	3412
decaf::util::logging::ConsoleHandler	1300
activemq::cmsutil::CmsAccessor	1068
activemq::cmsutil::CmsDestinationAccessor	1071
activemq::cmsutil::CmsTemplate	1083
cms::CMSException	1074
cms::CMSSecurityException	1082
cms::IllegalStateException	1868
cms::InvalidClientIdException	1992
cms::InvalidDestinationException	1993
cms::InvalidSelectorException	1999
cms::MessageEOFException	2492
cms::MessageFormatException	2493
cms::MessageNotReadableException	2548
cms::MessageNotWriteableException	2549
cms::UnsupportedOperationException	3659
activemq::util::CMSExceptionSupport	1077
cms::CMSProperties	1079
activemq::util::ActiveMQProperties	431
code	1096
activemq::state::CommandVisitor	1113
activemq::state::CommandVisitorAdapter	1120

activemq::state::ConnectionStateTracker	1293
decaf::lang::Comparable< T >	1125
decaf::lang::Comparable< bool >	1125
decaf::lang::Boolean	781
decaf::lang::Comparable< Boolean >	1125
decaf::lang::Boolean	781
decaf::lang::Comparable< BrokerId >	1125
activemq::commands::BrokerId	798
decaf::lang::Comparable< Byte >	1125
decaf::lang::Byte	884
decaf::lang::Comparable< ByteBuffer >	1125
decaf::nio::ByteBuffer	954
decaf::lang::Comparable< char >	1125
decaf::lang::Character	1019
decaf::lang::Comparable< Character >	1125
decaf::lang::Character	1019
decaf::lang::Comparable< CharBuffer >	1125
decaf::nio::CharBuffer	1037
decaf::lang::Comparable< ConnectionId >	1125
activemq::commands::ConnectionId	1231
decaf::lang::Comparable< ConsumerId >	1125
activemq::commands::ConsumerId	1330
decaf::lang::Comparable< Date >	1125
decaf::util::Date	1559
decaf::lang::Comparable< Delayed >	1125
decaf::util::concurrent::Delayed	1608
decaf::lang::Comparable< Double >	1125
decaf::lang::Double	1672
decaf::lang::Comparable< double >	1125
decaf::lang::Double	1672
decaf::lang::Comparable< DoubleBuffer >	1125
decaf::nio::DoubleBuffer	1692
decaf::lang::Comparable< float >	1125
decaf::lang::Float	1780
decaf::lang::Comparable< Float >	1125
decaf::lang::Float	1780
decaf::lang::Comparable< FloatBuffer >	1125
decaf::nio::FloatBuffer	1800
decaf::lang::Comparable< int >	1125
decaf::lang::Integer	1941
decaf::lang::Comparable< IntBuffer >	1125
decaf::nio::IntBuffer	1931
decaf::lang::Comparable< Integer >	1125
decaf::lang::Integer	1941
decaf::lang::Comparable< Level >	1125
decaf::util::logging::Level	2185
decaf::lang::Comparable< LocalTransactionId >	1125

activemq::commands::LocalTransactionId	2200
decaf::lang::Comparable< Long >	1125
decaf::lang::Long	2267
decaf::lang::Comparable< long long >	1125
decaf::lang::Long	2267
decaf::lang::Comparable< LongBuffer >	1125
decaf::nio::LongBuffer	2291
decaf::lang::Comparable< MessageId >	1125
activemq::commands::MessageId	2494
decaf::lang::Comparable< ProducerId >	1125
activemq::commands::ProducerId	2869
decaf::lang::Comparable< SessionId >	1125
activemq::commands::SessionId	3161
decaf::lang::Comparable< Short >	1125
decaf::lang::Short	3220
decaf::lang::Comparable< short >	1125
decaf::lang::Short	3220
decaf::lang::Comparable< ShortBuffer >	1125
decaf::nio::ShortBuffer	3238
decaf::lang::Comparable< TimeUnit >	1125
decaf::util::concurrent::TimeUnit	3559
decaf::lang::Comparable< TransactionId >	1125
activemq::commands::TransactionId	3569
activemq::commands::LocalTransactionId	2200
activemq::commands::XATransactionId	3765
decaf::lang::Comparable< unsigned char >	1125
decaf::lang::Byte	884
decaf::lang::Comparable< URI >	1125
decaf::net::URI	3660
decaf::lang::Comparable< UUID >	1125
decaf::util::UUID	3706
decaf::lang::Comparable< XATransactionId >	1125
activemq::commands::XATransactionId	3765
decaf::util::Comparator< T >	1127
activemq::util::CompositeData	1129
decaf::util::concurrent::locks::Condition	1156
decaf::util::concurrent::ConditionHandle	1162
decaf::internal::util::concurrent::ConditionImpl	1163
cms::ConnectionFactory	1228
activemq::core::ActiveMQConnectionFactory	251
cms::ConnectionMetaData	1287
activemq::core::ActiveMQConnectionMetaData	262
activemq::state::ConnectionState	1291
activemq::state::ConsumerState	1389
decaf::util::concurrent::CountDownLatch	1417
ct_data_s	1422
decaf::io::DataInput	1452
decaf::io::DataOutput	1468

activemq::wireformat::openwire::marshal::DataStreamMarshaller	1503
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller	741
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller . . .	293
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller	445
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller	530
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	556
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	588
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller	644
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	714
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	839
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller . .	1184
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller . . .	1216
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	1275
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller . . .	1318
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	1377
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller . . .	1405
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	1630
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	1830
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	2145
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	2415
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller . . .	2454
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	2482
activemq::wireformat::openwire::marshal::v1::MessageMarshaller	2540
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	174
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	213
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	332
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller . .	359
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller	403
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	503
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	617
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	2584
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	2862
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	2910
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	3003
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	3019
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller . . .	3050
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	3102
activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller	1438
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller . . .	1499
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	1742
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller . .	1974
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	3199
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	3260
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller . . .	3602
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	808
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	1245
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	1345
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller	1662
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	2036
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	2065
activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller	2087
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller . . .	2118
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	2518

activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller . . .	2636
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	2752
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller . .	2178
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	2893
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	3184
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	3442
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	3576
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller . .	2224
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller . . .	3781
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	3743
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller . . .	305
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	457
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller	541
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller	568
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	596
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	656
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	734
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	851
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller . .	1196
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller . . .	1204
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	1263
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller . . .	1306
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	1366
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller . . .	1393
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	1618
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	1818
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	2129
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	2403
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller . . .	2438
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	2470
activemq::wireformat::openwire::marshal::v2::MessageMarshaller	2532
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller	182
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller	229
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller	344
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller .	371
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	415
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller	515
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	629
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller	2568
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	2842
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	2906
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	2991
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller	3027
activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller . . .	3054
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	3089
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller	1426
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller . . .	1487
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	1726
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller . .	1962
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	3207
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	3256
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	3618
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	820

activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	1234
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	1334
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	1651
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	2021
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	2049
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller	2072
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	2102
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2499
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller	2617
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	2735
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	2166
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	2874
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	3165
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	3457
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller	3580
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	2208
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	3773
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	3735
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	289
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	441
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	527
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	552
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	580
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	636
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	701
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	831
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	1177
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	1208
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	1267
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller	1310
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	1369
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller	1397
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	1622
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	1822
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	2133
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	2407
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2442
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	2474
activemq::wireformat::openwire::marshal::v3::MessageMarshaller	2527
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller	171
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller	210
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	328
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	355
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	399
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	499
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	609
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	2576
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	2850
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	2918
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	2999
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller	3023
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	3058
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	3098

activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller	1430
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller . . .	1491
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	1730
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller . .	1966
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	3203
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	3268
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	3606
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	801
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	1238
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	1338
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller	1654
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	2028
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	2053
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller	2076
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	2106
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	2510
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller . . .	2628
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	2744
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller .	2162
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	2881
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	3180
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	3438
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	3583
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller . .	2212
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller . . .	3785
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	3747
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller . . .	297
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	449
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	534
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	560
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	584
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	640
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	708
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	835
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller . .	1180
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller . . .	1212
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1271
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller . . .	1314
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1373
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller . . .	1401
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	1626
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	1826
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	2137
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	2411
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller . . .	2450
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller	2478
activemq::wireformat::openwire::marshal::v4::MessageMarshaller	2536
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	178
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller	217
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	336
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller .	363
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	407
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	507

activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	613
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2580
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	2846
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	2902
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller	3011
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller	3039
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller . . .	3046
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	3085
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller	1434
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller . . .	1495
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	1738
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller . .	1970
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	3211
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	3272
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	3614
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	805
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	1241
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	1342
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller	1658
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	2032
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	2061
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller	2083
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	2114
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	2503
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller . . .	2632
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	2748
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller .	2174
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	2878
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	3169
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	3450
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	3587
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller . .	2220
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller . . .	3777
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	3739
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller . . .	301
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	453
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	538
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	564
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	592
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	648
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	721
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	843
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller . .	1188
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller . . .	1220
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	1279
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller . . .	1322
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller	1381
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller . . .	1409
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller	1638
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	1834
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller	2141
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller	2419
activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller . . .	2446

activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	2486
activemq::wireformat::openwire::marshal::v5::MessageMarshaller	2523
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	186
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	221
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	340
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller .	367
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	411
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	511
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	621
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller	2572
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller	2854
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	2914
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller	3007
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	3035
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller . . .	3066
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	3093
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller	1442
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller . . .	1479
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	1734
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller . .	1978
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	3195
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	3264
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	3598
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	812
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	1249
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller	1349
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller	1666
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	2025
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller	2045
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller	2091
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller	2110
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller	2507
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller . . .	2624
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	2739
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller .	2170
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	2885
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	3176
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	3446
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller	3572
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller . .	2216
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller . . .	3789
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	3728
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller . . .	309
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller	461
activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller	545
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	572
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	600
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller	652
activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller	728
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller	847
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller . .	1192
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller . . .	1224
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller	1283

activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller . . .	1326
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller	1385
activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller . . .	1413
activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller	1634
activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller	1814
activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller	2125
activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller	2399
activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller . . .	2458
activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller	2466
activemq::wireformat::openwire::marshal::v6::MessageMarshaller	2544
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller	190
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller	225
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	348
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller .	375
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	419
activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller	519
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	625
activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller	2588
activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller	2858
activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller	2922
activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller	2995
activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller	3031
activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller . . .	3062
activemq::wireformat::openwire::marshal::v6::ResponseMarshaller	3107
activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller	1446
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller . . .	1483
activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller	1722
activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller .	1958
activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller	3192
activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller	3252
activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller . . .	3610
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	816
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller	1253
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	1353
activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller	1647
activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller . . .	2017
activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller . . .	2057
activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller	2079
activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller . .	2099
activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller	2514
activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller .	2620
activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller	2731
activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller	2158
activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller	2889
activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller	3173
activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller . . .	3453
activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller	3591
activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller	2204
activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller .	3769
activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller	3731
decaf::internal::net::ssl::DefaultSSLContext	1581
decaf::util::zip::Deflater	1595
cms::DeliveryMode	1609

cms::Destination	1610
cms::Queue	2947
activemq::commands::ActiveMQQueue	435
cms::TemporaryQueue	3516
activemq::commands::ActiveMQTempQueue	549
cms::TemporaryTopic	3517
activemq::commands::ActiveMQTempTopic	576
cms::Topic	3568
activemq::commands::ActiveMQTopic	633
activemq::commands::ActiveMQDestination::DestinationFilter	1613
activemq::cmsutil::DestinationResolver	1642
activemq::cmsutil::DynamicDestinationResolver	1704
activemq::core::DispatchData	1670
activemq::core::Dispatcher	1671
activemq::core::ActiveMQConsumer	268
activemq::core::ActiveMQSession	465
decaf::lang::DYNAMIC_CAST_TOKEN	1704
decaf::util::Map< K, V, COMPARATOR >::Entry	1706
decaf::util::logging::ErrorManager	1710
cms::ExceptionListener	1719
decaf::util::concurrent::Executor	1749
decaf::util::concurrent::ExecutorService	1751
decaf::io::FileDescriptor	1768
decaf::internal::net::SocketFileDescriptor	3305
decaf::util::logging::Filter	1770
decaf::io::Flushable	1811
decaf::io::OutputStream	2718
decaf::io::Writer	3756
decaf::util::logging::Formatter	1838
decaf::util::logging::SimpleFormatter	3278
decaf::util::logging::XMLFormatter	3793
decaf::util::concurrent::Future< V >	1840
activemq::transport::correlator::FutureResponse	1843
gz_header_s	1848
gz_state	1849
decaf::internal::util::HexStringParser	1856
activemq::wireformat::openwire::utils::HexTable	1857
activemq::util::IdGenerator	1861
decaf::net::InetAddress	1884
decaf::net::Inet4Address	1880
decaf::net::Inet6Address	1883
inflate_state	1892
decaf::util::zip::Inflater	1894
internal_state	1982
decaf::lang::Iterable< E >	2011
decaf::util::Collection< E >	1097
decaf::util::AbstractCollection< E >	143
decaf::util::List< E >	2190
decaf::util::AbstractList< E >	156
decaf::util::AbstractSequentialList< E >	161

decaf::util::StlList< E >	3357
decaf::util::Queue< E >	2948
decaf::util::AbstractQueue< E >	158
decaf::util::concurrent::BlockingQueue< E >	774
decaf::util::concurrent::SynchronousQueue< E >	3477
decaf::util::PriorityQueue< E >	2832
decaf::util::Set< E >	3220
decaf::util::AbstractSet< E >	162
decaf::util::StlSet< E >	3390
decaf::lang::Iterable< PrimitiveValueNode >	2011
decaf::util::Collection< PrimitiveValueNode >	1097
decaf::util::AbstractCollection< PrimitiveValueNode >	143
decaf::util::List< PrimitiveValueNode >	2190
decaf::util::StlList< PrimitiveValueNode >	3357
activemq::util::PrimitiveList	2787
decaf::util::Iterator< T >	2012
decaf::util::Iterator< E >	2012
decaf::util::ListIterator< E >	2197
decaf::security::Key	2149
decaf::security::PublicKey	2940
decaf::util::concurrent::Lock	2228
decaf::util::concurrent::locks::Lock	2229
decaf::util::concurrent::locks::ReentrantLock	2977
decaf::util::concurrent::locks::LockSupport	2234
decaf::util::logging::Logger	2237
decaf::util::logging::LoggerHierarchy	2249
decaf::util::logging::LogManager	2254
decaf::util::logging::LogRecord	2261
decaf::util::logging::LogWriter	2266
activemq::util::LongSequenceGenerator	2302
decaf::util::logging::MarkBlockLogger	2327
activemq::wireformat::MarshalAware	2328
activemq::commands::DataStructure	1553
activemq::commands::BaseDataStructure	764
activemq::commands::ActiveMQDestination	279
activemq::commands::ActiveMQQueue	435
activemq::commands::ActiveMQTempDestination	523
activemq::commands::ActiveMQTopic	633
activemq::commands::BooleanExpression	786
activemq::commands::BrokerId	798
activemq::commands::Command	1107
activemq::commands::BaseCommand	694
activemq::commands::BrokerError	793
activemq::commands::BrokerInfo	824
activemq::commands::ConnectionControl	1172
activemq::commands::ConnectionError	1200
activemq::commands::ConnectionInfo	1257
activemq::commands::ConsumerControl	1302
activemq::commands::ConsumerInfo	1357
activemq::commands::ControlCommand	1390
activemq::commands::DestinationInfo	1613

activemq::commands::FlushCommand	1812
activemq::commands::KeepAliveInfo	2122
activemq::commands::Message	2358
activemq::commands::ActiveMQMessageTemplate< T >	379
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >	379
activemq::commands::ActiveMQBytesMessage	194
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >	379
activemq::commands::ActiveMQMapMessage	316
activemq::commands::ActiveMQMessageTemplate< cms::Message >	379
activemq::commands::ActiveMQBlobMessage	166
activemq::commands::ActiveMQMessage	352
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >	379
activemq::commands::ActiveMQObjectMessage	396
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >	379
activemq::commands::ActiveMQStreamMessage	485
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >	379
activemq::commands::ActiveMQTextMessage	604
activemq::commands::MessageAck	2394
activemq::commands::MessageDispatch	2427
activemq::commands::MessageDispatchNotification	2462
activemq::commands::MessagePull	2563
activemq::commands::ProducerAck	2839
activemq::commands::ProducerInfo	2897
activemq::commands::RemoveInfo	2988
activemq::commands::RemoveSubscriptionInfo	3015
activemq::commands::ReplayCommand	3043
activemq::commands::Response	3076
activemq::commands::DataArrayResponse	1423
activemq::commands::DataResponse	1476
activemq::commands::ExceptionResponse	1720
activemq::commands::IntegerResponse	1956
activemq::state::Tracked	3569
activemq::commands::SessionInfo	3188
activemq::commands::ShutdownInfo	3249
activemq::commands::TransactionInfo	3594
activemq::commands::WireFormatInfo	3717
activemq::commands::ConnectionId	1231
activemq::commands::ConsumerId	1330
activemq::commands::DiscoveryEvent	1643
activemq::commands::JournalQueueAck	2014
activemq::commands::JournalTopicAck	2040
activemq::commands::JournalTrace	2069
activemq::commands::JournalTransaction	2095
activemq::commands::MessageId	2494
activemq::commands::NetworkBridgeFilter	2613
activemq::commands::PartialCommand	2727
activemq::commands::LastPartialCommand	2156
activemq::commands::ProducerId	2869
activemq::commands::SessionId	3161
activemq::commands::SubscriptionInfo	3433
activemq::commands::TransactionId	3569

activemq::wireformat::openwire::marshal::v6::MarshallerFactory	2331
activemq::wireformat::openwire::marshal::v3::MarshallerFactory	2331
activemq::wireformat::openwire::marshal::v4::MarshallerFactory	2332
activemq::wireformat::openwire::marshal::v5::MarshallerFactory	2333
activemq::wireformat::openwire::marshal::v1::MarshallerFactory	2334
activemq::wireformat::openwire::marshal::v2::MarshallerFactory	2334
activemq::util::MarshallingSupport	2335
decaf::lang::Math	2339
cms::Message	2375
activemq::commands::ActiveMQMessageTemplate< cms::Message >	379
cms::BytesMessage	979
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >	379
cms::MapMessage	2318
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >	379
cms::ObjectMessage	2658
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >	379
cms::StreamMessage	3415
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >	379
cms::TextMessage	3519
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >	379
activemq::cmsutil::MessageCreator	2426
cms::MessageEnumeration	2490
activemq::core::ActiveMQQueueBrowser	438
cms::MessageListener	2522
activemq::wireformat::openwire::utils::MessagePropertyInterceptor	2557
decaf::util::concurrent::MutexHandle	2609
decaf::internal::util::concurrent::MutexImpl	2609
decaf::internal::net::Network	2611
decaf::lang::Number	2653
decaf::lang::Byte	884
decaf::lang::Character	1019
decaf::lang::Double	1672
decaf::lang::Float	1780
decaf::lang::Integer	1941
decaf::lang::Long	2267
decaf::lang::Short	3220
decaf::util::concurrent::atomic::AtomicInteger	680
decaf::internal::net::ssl::openssl::OpenSSLParameters	2661
decaf::lang::Pointer< T, REFCOUNTER >	2756
decaf::util::concurrent::PooledThreadListener	2779
decaf::util::concurrent::ThreadPool	3531
activemq::core::PrefetchPolicy	2783
activemq::core::policies::DefaultPrefetchPolicy	1565
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller	2808
activemq::util::PrimitiveValueNode::PrimitiveValue	2814
activemq::util::PrimitiveValueConverter	2816
activemq::util::PrimitiveValueNode	2817
decaf::security::Principal	2831
decaf::security::auth::x500::X500Principal	3761
activemq::cmsutil::ProducerCallback	2866
activemq::cmsutil::CmsTemplate::SendExecutor	3135

activemq::state::ProducerState	2926
decaf::util::Properties	2927
decaf::util::logging::PropertiesChangeListener	2936
decaf::util::Random	2953
decaf::security::SecureRandom	3116
decaf::lang::Readable	2958
decaf::io::Reader	2960
decaf::util::concurrent::locks::ReadWriteLock	2968
activemq::core::RedeliveryPolicy	2972
activemq::core::policies::DefaultRedeliveryPolicy	1569
decaf::util::concurrent::RejectedExecutionHandler	2987
decaf::internal::util::Resource	3072
decaf::internal::util::GenericResource< T >	1847
decaf::internal::util::ResourceLifecycleManager	3073
activemq::cmsutil::ResourceLifecycleManager	3074
activemq::transport::mock::ResponseBuilder	3079
activemq::wireformat::openwire::OpenWireResponseBuilder	2717
decaf::lang::Runnable	3111
activemq::threads::CompositeTaskRunner	1133
activemq::threads::DedicatedTaskRunner	1564
activemq::transport::IOTransport	2005
decaf::lang::Thread	3520
activemq::transport::mock::InternalCommandListener	1986
decaf::util::concurrent::PooledThread	2777
decaf::util::TimerTask	3554
activemq::transport::inactivity::ReadChecker	2959
activemq::transport::inactivity::WriteChecker	3755
decaf::lang::Runtime	3112
decaf::internal::DecafRuntime	1563
decaf::security::SecureRandomSpi	3124
decaf::internal::security::SecureRandomImpl	3121
decaf::internal::security::SecureRandomImpl	3121
decaf::util::concurrent::Semaphore	3126
decaf::net::ServerSocket	3136
decaf::net::ssl::SSLServerSocket	3329
decaf::internal::net::ssl::openssl::OpenSSLServerSocket	2664
decaf::net::ServerSocketFactory	3145
decaf::internal::net::DefaultServerSocketFactory	1574
decaf::net::ssl::SSLServerSocketFactory	3335
decaf::internal::net::ssl::DefaultSSLServerSocketFactory	1582
decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory	2669
activemq::cmsutil::SessionCallback	3160
activemq::cmsutil::CmsTemplate::ProducerExecutor	2867
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor	3070
activemq::cmsutil::CmsTemplate::ReceiveExecutor	2970
activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor	3071
activemq::cmsutil::SessionPool	3215
activemq::state::SessionState	3217
decaf::util::logging::SimpleLogger	3279
decaf::net::SocketAddress	3297

decaf::net::InetSocketAddress	1891
decaf::net::SocketError	3298
decaf::net::SocketFactory	3301
decaf::internal::net::DefaultSocketFactory	1577
decaf::net::ssl::SSLSocketFactory	3345
decaf::internal::net::ssl::DefaultSSLSocketFactory	1587
decaf::internal::net::ssl::openssl::OpenSSLSocketFactory	2689
decaf::net::SocketImplFactory	3314
decaf::net::SocketOptions	3315
decaf::net::SocketImpl	3306
decaf::internal::net::tcp::TcpSocket	3497
decaf::net::ssl::SSLContext	3321
decaf::net::ssl::SSLContextSpi	3324
decaf::internal::net::ssl::openssl::OpenSSLContextSpi	2658
decaf::net::ssl::SSLParameters	3326
activemq::commands::BrokerError::StackTraceElement	3350
cms::Startable	3355
cms::Connection	1168
decaf::lang::STATIC_CAST_TOKEN	3356
activemq::core::ActiveMQConstants::StaticInitializer	3356
activemq::wireformat::stomp::StompCommandConstants	3395
activemq::wireformat::stomp::StompFrame	3398
activemq::wireformat::stomp::StompHelper	3402
cms::Stoppable	3411
cms::Connection	1168
decaf::util::StringTokenizer	3430
decaf::util::concurrent::Synchronizable	3461
activemq::core::MessageDispatchChannel	2431
decaf::util::Collection< PrimitiveValueNode >	1097
decaf::internal::util::concurrent::SynchronizableImpl	3473
decaf::io::InputStream	1909
decaf::io::OutputStream	2718
decaf::util::Collection< E >	1097
decaf::util::concurrent::Mutex	2604
decaf::util::Map< K, V, COMPARATOR >	2305
decaf::util::AbstractMap< K, V, COMPARATOR >	157
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >	1136
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >	1140
decaf::util::StlMap< K, V, COMPARATOR >	3369
decaf::util::StlQueue< T >	3382
decaf::util::Map< std::string, PrimitiveValueNode, std::less< std::string > >	2305
decaf::util::StlMap< std::string, PrimitiveValueNode >	3369
activemq::util::PrimitiveMap	2798
activemq::core::Synchronization	3477
decaf::lang::System	3487
activemq::threads::Task	3494
activemq::core::ActiveMQSessionExecutor	482
activemq::threads::CompositeTask	1132
activemq::transport::failover::BackupTransportPool	692
activemq::transport::failover::CloseTransportsTask	1066
activemq::transport::failover::FailoverTransport	1752

activemq::threads::CompositeTaskRunner	1133
decaf::util::concurrent::TaskListener	3495
activemq::threads::TaskRunner	3496
activemq::threads::CompositeTaskRunner	1133
activemq::threads::DedicatedTaskRunner	1564
decaf::util::concurrent::ThreadFactory	3529
decaf::lang::ThreadGroup	3531
decaf::lang::Throwable	3536
decaf::lang::Exception	1712
activemq::exceptions::ActiveMQException	313
activemq::exceptions::BrokerException	797
decaf::io::IOException	2003
decaf::io::EOFException	1707
decaf::io::InterruptedIOException	1990
decaf::net::SocketTimeoutException	3319
decaf::io::UnsupportedEncodingException	3654
decaf::io::UTFDataFormatException	3703
decaf::net::HttpRetryException	1858
decaf::net::MalformedURLException	2303
decaf::net::ProtocolException	2937
decaf::net::SocketException	3298
decaf::internal::net::ssl::openssl::OpenSSLSocketException	2686
decaf::net::BindException	768
decaf::net::ConnectException	1165
decaf::net::NoRouteToHostException	2640
decaf::net::PortUnreachableException	2781
decaf::net::UnknownHostException	3649
decaf::net::UnknownServiceException	3652
decaf::util::zip::ZipException	3796
decaf::lang::exceptions::ClassCastException	1062
decaf::lang::exceptions::IllegalArgumentException	1863
decaf::lang::exceptions::IllegalMonitorStateException	1865
decaf::lang::exceptions::IllegalStateException	1869
decaf::nio::InvalidMarkException	1997
decaf::lang::exceptions::IllegalThreadStateException	1872
decaf::lang::exceptions::IndexOutOfBoundsException	1877
decaf::lang::exceptions::InterruptedException	1987
decaf::lang::exceptions::InvalidStateException	2000
decaf::lang::exceptions::NoSuchElementException	2645
decaf::lang::exceptions::NullPointerException	2650
decaf::lang::exceptions::NumberFormatException	2655
decaf::lang::exceptions::RuntimeException	3114
decaf::lang::exceptions::UnsupportedOperationException	3657
decaf::nio::ReadOnlyBufferException	2966
decaf::net::URISyntaxException	3686
decaf::nio::BufferOverflowException	880
decaf::nio::BufferUnderflowException	882
decaf::security::GeneralSecurityException	1845
decaf::security::cert::CertificateException	1012
decaf::security::cert::CertificateEncodingException	1010
decaf::security::cert::CertificateExpiredException	1014
decaf::security::cert::CertificateNotYetValidException	1015

decaf::security::cert::CertificateParsingException	1017
decaf::security::KeyException	2151
decaf::security::InvalidKeyException	1994
decaf::security::KeyManagementException	2153
decaf::security::NoSuchAlgorithmException	2643
decaf::security::NoSuchProviderException	2648
decaf::security::SignatureException	3276
decaf::util::concurrent::BrokenBarrierException	790
decaf::util::concurrent::CancellationException	1004
decaf::util::concurrent::ExecutionException	1746
decaf::util::concurrent::RejectedExecutionException	2984
decaf::util::concurrent::TimeoutException	3541
decaf::util::zip::DataFormatException	1450
decaf::util::Timer	3543
decaf::internal::util::TimerTaskHeap	3556
activemq::state::TransactionState	3622
decaf::internal::util::concurrent::Transferer< E >	3625
decaf::internal::util::concurrent::TransferQueue< E >	3625
decaf::internal::util::concurrent::TransferStack< E >	3627
activemq::transport::TransportFactory	3634
activemq::transport::AbstractTransportFactory	164
activemq::transport::failover::FailoverTransportFactory	1763
activemq::transport::mock::MockTransportFactory	2602
activemq::transport::tcp::TcpTransportFactory	3514
activemq::transport::tcp::SslTransportFactory	3349
activemq::transport::TransportListener	3643
activemq::core::ActiveMQConnection	233
activemq::transport::DefaultTransportListener	1593
activemq::transport::failover::BackupTransport	690
activemq::transport::mock::InternalCommandListener	1986
activemq::transport::failover::FailoverTransportListener	1766
activemq::transport::TransportFilter	3636
activemq::transport::TransportRegistry	3645
tree_desc_s	3648
decaf::lang::Thread::UncaughtExceptionHandler	3648
decaf::internal::net::URIEncoderDecoder	3672
decaf::internal::net::URIHelper	3674
activemq::transport::failover::URIPool	3681
activemq::util::URISupport	3684
decaf::internal::net::URIType	3690
decaf::net::URL	3697
decaf::net::URLDecoder	3699
decaf::net::URLEncoder	3700
activemq::util::Usage	3701
activemq::util::MemoryUsage	2354
activemq::wireformat::WireFormat	3712
activemq::wireformat::openwire::OpenWireFormat	2700
activemq::wireformat::stomp::StompWireFormat	3407
activemq::wireformat::WireFormatFactory	3716
activemq::wireformat::openwire::OpenWireFormatFactory	2712
activemq::wireformat::stomp::StompWireFormatFactory	3410

activemq::wireformat::WireFormatRegistry	3752
z_stream_s	3795

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

decaf::util::AbstractCollection < E > (This class provides a skeletal implementation of the Collection (p. 1097) interface, to minimize the effort required to implement this interface)	143
decaf::util::AbstractList < E > (This class provides a skeletal implementation of the List (p. 2190) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array))	156
decaf::util::AbstractMap < K , V , COMPARATOR > (This class provides a skeletal implementation of the Map (p. 2305) interface, to minimize the effort required to implement this interface)	157
decaf::util::AbstractQueue < E > (This class provides skeletal implementations of some Queue (p. 2948) operations)	158
decaf::util::AbstractSequentialList < E > (This class provides a skeletal implementation of the List (p. 2190) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list))	161
decaf::util::AbstractSet < E > (This class provides a skeletal implementation of the Set (p. 3220) interface to minimize the effort required to implement this interface)	162
activemq::transport::AbstractTransportFactory (Abstract implementation of the TransportFactory (p. 3634) interface, providing the base functionality that's common to most of the TransportFactory (p. 3634) instances)	164
activemq::core::ActiveMQAckHandler (Interface class that is used to give CMS Messages an interface to Ack themselves with)	165
activemq::commands::ActiveMQBlobMessage	166
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 171))	171
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 174))	174
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 178))	178

activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 182))	182
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 186))	186
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 190))	190
activemq::commands::ActiveMQBytesMessage	194
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 210))	210
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 213))	213
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 217))	217
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 221))	221
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 225))	225
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 229))	229
activemq::core::ActiveMQConnection (Concrete connection used for all connectors to the ActiveMQ broker)	233
activemq::core::ActiveMQConnectionFactory	251
activemq::core::ActiveMQConnectionMetaData (This class houses all the various settings and information that is used by an instance of an ActiveMQConnection (p. 233) class)	262
activemq::core::ActiveMQConstants (Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values)	265
activemq::core::ActiveMQConsumer	268
activemq::library::ActiveMQCPP	277
activemq::commands::ActiveMQDestination	279
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 289))	289
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 293))	293
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 297))	297
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 301))	301

activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 305))	305
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 309))	309
activemq::exceptions::ActiveMQException	313
activemq::commands::ActiveMQMapMessage	316
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 328))	328
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 332))	332
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 336))	336
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 340))	340
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 344))	344
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 348))	348
activemq::commands::ActiveMQMessage	352
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 355))	355
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 359))	359
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 363))	363
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 367))	367
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 371))	371
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 375))	375
activemq::commands::ActiveMQMessageTemplate< T >	379
activemq::commands::ActiveMQObjectMessage	396
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 399))	399
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 403))	403

activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 407))	407
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 411))	411
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 415))	415
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 419))	419
activemq::core::ActiveMQProducer	423
activemq::util::ActiveMQProperties (Implementation of the CMSProperties interface that delegates to a decaf::util::Properties (p. 2927) object)	431
activemq::commands::ActiveMQQueue	435
activemq::core::ActiveMQQueueBrowser	438
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 441))	441
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 445))	445
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 449))	449
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 453))	453
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 457))	457
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 461))	461
activemq::core::ActiveMQSession	465
activemq::core::ActiveMQSessionExecutor (Delegate dispatcher for a single session)	482
activemq::commands::ActiveMQStreamMessage	485
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller (p. 499))	499
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller (p. 503))	503
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller (p. 507))	507
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller (p. 511))	511
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller (p. 515))	515

activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessage- Marshaller (p. 519))	519
activemq::commands::ActiveMQTempDestination	523
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 527))	527
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 530))	530
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 534))	534
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 538))	538
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 541))	541
activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 545))	545
activemq::commands::ActiveMQTempQueue	549
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMar- shaller (p. 552))	552
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMar- shaller (p. 556))	556
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMar- shaller (p. 560))	560
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMar- shaller (p. 564))	564
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMar- shaller (p. 568))	568
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMar- shaller (p. 572))	572
activemq::commands::ActiveMQTempTopic	576
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMar- shaller (p. 580))	580
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMar- shaller (p. 584))	584
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMar- shaller (p. 588))	588
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMar- shaller (p. 592))	592

activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 596))	596
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 600))	600
activemq::commands::ActiveMQTextMessage	604
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 609))	609
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 613))	613
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 617))	617
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 621))	621
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 625))	625
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 629))	629
activemq::commands::ActiveMQTopic	633
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 636))	636
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 640))	640
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 644))	644
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 648))	648
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 652))	652
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 656))	656
activemq::core::ActiveMQTransactionContext (Transaction Management class, hold messages that are to be redelivered upon a request to roll-back)	660
decaf::util::zip::Adler32 (Clas that can be used to compute an Adler-32 Checksum (p. 1059) for a data stream)	663
decaf::lang::Appendable (An object to which char sequences and values can be appended)	666
decaf::internal::AprPool (Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made)	668

decaf::lang::ArrayPointer< T, REFCOUNTER > (Decaf's implementation of a Smart Pointer (p. 2756) that is a template on a Type and is Thread (p. 3520) Safe if the default Reference Counter is used)	669
decaf::lang::ArrayPointerComparator< T, R > (This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this ArrayPointer (p. 669))	676
decaf::util::concurrent::atomic::AtomicBoolean (A boolean value that may be updated atomically)	677
decaf::util::concurrent::atomic::AtomicInteger (An int value that may be updated atomically)	680
decaf::util::concurrent::atomic::AtomicRefCounter	685
decaf::util::concurrent::atomic::AtomicReference< T > (An Pointer reference that may be updated atomically)	687
activemq::transport::failover::BackupTransport	690
activemq::transport::failover::BackupTransportPool	692
activemq::commands::BaseCommand	694
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 701))	701
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 708))	708
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 714))	714
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 721))	721
activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 728))	728
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 734))	734
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller (Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol)	741
activemq::commands::BaseDataStructure	764
binary_function	768
decaf::net::BindException	768
decaf::io::BlockingByteArrayInputStream (This is a blocking version of a byte buffer stream)	771
decaf::util::concurrent::BlockingQueue< E > (A decaf::util::Queue (p. 2948) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element)	774
decaf::lang::Boolean	781
activemq::commands::BooleanExpression	786
activemq::wireformat::openwire::utils::BooleanStream (Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream)	787
decaf::util::concurrent::BrokenBarrierException	790

activemq::commands::BrokerError (This class represents an Exception sent from the Broker)	793
activemq::exceptions::BrokerException	797
activemq::commands::BrokerId	798
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 801))	801
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 805))	805
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 808))	808
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 812))	812
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 816))	816
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 820))	820
activemq::commands::BrokerInfo	824
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 831))	831
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 835))	835
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 839))	839
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 843))	843
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 847))	847
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 851))	851
decaf::nio::Buffer (A container for data of a specific primitive type)	855
decaf::io::BufferedInputStream (A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream)	861
decaf::io::BufferedOutputStream (Wrapper around another output stream that buffers output before writing to the target output stream)	867
decaf::internal::nio::BufferFactory (Factory class used by static methods in the decaf::nio (p. 132) package to create the various default version of the NIO interfaces)	869
decaf::nio::BufferOverflowException	880
decaf::nio::BufferUnderflowException	882
decaf::lang::Byte	884
decaf::internal::util::ByteArrayAdapter (This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data)	893
decaf::internal::nio::ByteArrayBuffer (This class defines six categories of operations upon byte buffers:)	915
decaf::io::ByteArrayInputStream (A ByteArrayInputStream (p. 944) contains an internal buffer that contains bytes that may be read from the stream)	944
decaf::io::ByteArrayOutputStream	951
decaf::nio::ByteBuffer (This class defines six categories of operations upon byte buffers:)	954
cms::BytesMessage (A BytesMessage (p. 979) object is used to send a message containing a stream of unsigned bytes)	979

activemq::cmsutil::CachedConsumer (A cached message consumer contained within a pooled session)	993
activemq::cmsutil::CachedProducer (A cached message producer contained within a pooled session)	996
decaf::util::concurrent::Callable< V > (A task that returns a result and may throw an exception)	1003
decaf::util::concurrent::CancellationException	1004
decaf::security::cert::Certificate (Base interface for all identity certificates)	1007
decaf::security::cert::CertificateEncodingException	1010
decaf::security::cert::CertificateException	1012
decaf::security::cert::CertificateExpiredException	1014
decaf::security::cert::CertificateNotYetValidException	1015
decaf::security::cert::CertificateParsingException	1017
decaf::lang::Character	1019
decaf::internal::nio::CharArrayBuffer	1027
decaf::nio::CharBuffer (This class defines four categories of operations upon character buffers:)	1037
decaf::lang::CharSequence (A CharSequence (p. 1053) is a readable sequence of char values)	1053
decaf::util::zip::CheckedInputStream (An implementation of a FilterInputStream that will maintain a Checksum (p. 1059) of the bytes read, the Checksum (p. 1059) can then be used to verify the integrity of the input stream)	1055
decaf::util::zip::CheckedOutputStream (An implementation of a FilterOutputStream that will maintain a Checksum (p. 1059) of the bytes written, the Checksum (p. 1059) can then be used to verify the integrity of the output stream)	1058
decaf::util::zip::Checksum (An interface used to represent Checksum (p. 1059) values in the Zip package)	1059
decaf::lang::exceptions::ClassCastException	1062
cms::Closeable (Interface for a class that implements the close method)	1064
decaf::io::Closeable (Interface for a class that implements the close method)	1065
activemq::transport::failover::CloseTransportsTask	1066
activemq::cmsutil::CmsAccessor (Base class for activemq::cmsutil.CmsTemplate (p. 1083) and other CMS-accessing gateway helpers, defining common properties such as the CMS cms.ConnectionFactory (p. 1228) to operate on)	1068
activemq::cmsutil::CmsDestinationAccessor (Extends the CmsAccessor (p. 1068) to add support for resolving destination names)	1071
cms::CMSException (CMS API Exception that is the base for all exceptions thrown from CMS classes)	1074
activemq::util::CMSExceptionSupport	1077
cms::CMSProperties (Interface for a Java-like properties object)	1079
cms::CMSSecurityException (This exception must be thrown when a provider rejects a user name/password submitted by a client)	1082
activemq::cmsutil::CmsTemplate (CmsTemplate (p. 1083) simplifies performing synchronous CMS operations)	1083
code	1096
decaf::util::Collection< E > (The root interface in the collection hierarchy)	1097
activemq::commands::Command	1107
activemq::state::CommandVisitor (Interface for an Object that can visit the various Command Objects that are sent from and to this client)	1113
activemq::state::CommandVisitorAdapter (Default Implementation of a CommandVisitor (p. 1113) that returns NULL for all calls)	1120
decaf::lang::Comparable< T > (This interface imposes a total ordering on the objects of each class that implements it)	1125

decaf::util::Comparator< T > (A comparison function, which imposes a total ordering on some collection of objects)	1127
activemq::util::CompositeData (Represents a Composite URI)	1129
activemq::threads::CompositeTask (Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p.1133))	1132
activemq::threads::CompositeTaskRunner (A Task (p.3494) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added)	1133
activemq::transport::CompositeTransport (A Composite Transport (p.3629) is a Transport (p.3629) implementation that is composed of several Transports) .	1135
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > (Interface for a Map (p.2305) type that provides additional atomic putIfAbsent , remove , and replace methods alongside the already available Map (p.2305) interface) .	1136
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (Map (p.2305) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map) . . .	1140
decaf::util::concurrent::locks::Condition (Condition (p.1156) factors out the Mutex (p.2604) monitor methods (wait , notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary Lock (p.2229) implementations)	1156
decaf::util::concurrent::ConditionHandle	1162
decaf::internal::util::concurrent::ConditionImpl	1163
decaf::net::ConnectException	1165
cms::Connection (The client's connection to its provider)	1168
activemq::commands::ConnectionControl	1172
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p.1177))	1177
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p.1180))	1180
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p.1184))	1184
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p.1188))	1188
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p.1192))	1192
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p.1196))	1196
activemq::commands::ConnectionError	1200
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p.1204))	1204
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p.1208))	1208
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p.1212))	1212

activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1216))	1216
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1220))	1220
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1224))	1224
cms::ConnectionFactory (Defines the interface for a factory that creates connection objects, the Connection (p. 1168) objects returned implement the CMS Con- nection (p. 1168) interface and hide the CMS Provider specific implementation details behind that interface)	1228
activemq::commands::ConnectionId	1231
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (Mar- shaling code for Open Wire Format for ConnectionIdMarshaller (p. 1234))	1234
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (Mar- shaling code for Open Wire Format for ConnectionIdMarshaller (p. 1238))	1238
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (Mar- shaling code for Open Wire Format for ConnectionIdMarshaller (p. 1241))	1241
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (Mar- shaling code for Open Wire Format for ConnectionIdMarshaller (p. 1245))	1245
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (Mar- shaling code for Open Wire Format for ConnectionIdMarshaller (p. 1249))	1249
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (Mar- shaling code for Open Wire Format for ConnectionIdMarshaller (p. 1253))	1253
activemq::commands::ConnectionInfo	1257
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1263))	1263
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1267))	1267
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1271))	1271
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1275))	1275
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1279))	1279
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1283))	1283
cms::ConnectionMetaData (A ConnectionMetaData (p. 1287) object provides in- formation describing the Connection (p. 1168) object)	1287
activemq::state::ConnectionState	1291

activemq::state::ConnectionStateTracker	1293
decaf::util::logging::ConsoleHandler (This Handler (p. 1852) publishes log records to System.err)	1300
activemq::commands::ConsumerControl	1302
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1306))	1306
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1310))	1310
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1314))	1314
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1318))	1318
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1322))	1322
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1326))	1326
activemq::commands::ConsumerId	1330
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1334))	1334
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1338))	1338
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1342))	1342
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1345))	1345
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1349))	1349
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1353))	1353
activemq::commands::ConsumerInfo	1357
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1366))	1366
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1369))	1369
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1373))	1373
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1377))	1377

activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1381))	1381
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1385))	1385
activemq::state::ConsumerState	1389
activemq::commands::ControlCommand	1390
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1393))	1393
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1397))	1397
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1401))	1401
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1405))	1405
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1409))	1409
activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1413))	1413
decaf::util::concurrent::CountDownLatch	1417
decaf::util::zip::CRC32 (Class that can be used to compute a CRC-32 checksum for a data stream)	1420
ct_data_s	1422
activemq::commands::DataArrayResponse	1423
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1426))	1426
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1430))	1430
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1434))	1434
activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1438))	1438
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1442))	1442
activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1446))	1446
decaf::util::zip::DataFormatException	1450
decaf::io::DataInput (The DataInput (p. 1452) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types)	1452

decaf::io::DataInputStream (A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way)	1460
decaf::io::DataOutput (The DataOutput (p. 1468) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream)	1468
decaf::io::DataOutputStream (A data output stream lets an application write primitive Java data types to an output stream in a portable way)	1473
activemq::commands::DataResponse	1476
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (Marshaling code for Open Wire Format for DataResponseMarshaller (p. 1479))	1479
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (Marshaling code for Open Wire Format for DataResponseMarshaller (p. 1483))	1483
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (Marshaling code for Open Wire Format for DataResponseMarshaller (p. 1487))	1487
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (Marshaling code for Open Wire Format for DataResponseMarshaller (p. 1491))	1491
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (Marshaling code for Open Wire Format for DataResponseMarshaller (p. 1495))	1495
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (Marshaling code for Open Wire Format for DataResponseMarshaller (p. 1499))	1499
activemq::wireformat::openwire::marshal::DataStreamMarshaller (Base class for all classes that marshal commands for Openwire)	1503
activemq::commands::DataStructure	1553
decaf::util::Date (Wrapper class around a time value in milliseconds)	1559
decaf::internal::DecafRuntime (Handles APR initialization and termination)	1563
activemq::threads::DedicatedTaskRunner	1564
activemq::core::policies::DefaultPrefetchPolicy	1565
activemq::core::policies::DefaultRedeliveryPolicy	1569
decaf::internal::net::DefaultServerSocketFactory (Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options)	1574
decaf::internal::net::DefaultSocketFactory (SocketFactory implementation that is used to create Sockets)	1577
decaf::internal::net::ssl::DefaultSSLContext (Default SSLContext manager for the Decaf library)	1581
decaf::internal::net::ssl::DefaultSSLServerSocketFactory (Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds)	1582
decaf::internal::net::ssl::DefaultSSLSocketFactory (Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds)	1587
activemq::transport::DefaultTransportListener	1593
decaf::util::zip::Deflater (This class compresses data using the <i>DEFLATE</i> algorithm (see specification))	1595
decaf::util::zip::DeflaterOutputStream (Provides a <i>FilterOutputStream</i> instance that compresses the data before writing it to the wrapped <i>OutputStream</i>) . .	1604

decaf::util::concurrent::Delayed (A mix-in style interface for marking objects that should be acted upon after a given delay)	1608
cms::DeliveryMode (This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages)	1609
cms::Destination (A Destination (p. 1610) object encapsulates a provider-specific address)	1610
activemq::commands::ActiveMQDestination::DestinationFilter	1613
activemq::commands::DestinationInfo	1613
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1618))	1618
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1622))	1622
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1626))	1626
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1630))	1630
activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1634))	1634
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1638))	1638
activemq::cmsutil::DestinationResolver (Resolves a CMS destination name to a Destination)	1642
activemq::commands::DiscoveryEvent	1643
activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1647))	1647
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1651))	1651
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1654))	1654
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1658))	1658
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1662))	1662
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1666))	1666
activemq::core::DispatchData (Simple POJO that contains the information necessary to route a message to a specified consumer)	1670
activemq::core::Dispatcher (Interface for an object responsible for dispatching messages to consumers)	1671
decaf::lang::Double	1672
decaf::internal::nio::DoubleArrayBuffer	1683

decaf::nio::DoubleBuffer (This class defines four categories of operations upon double buffers:)	1692
decaf::lang::DYNAMIC_CAST_TOKEN	1704
activemq::cmsutil::DynamicDestinationResolver (Resolves a CMS destination name to a Destination)	1704
decaf::util::Map< K, V, COMPARATOR >::Entry	1706
decaf::io::EOFException	1707
decaf::util::logging::ErrorManager (ErrorManager (p.1710) objects can be attached to Handlers to process any error that occur on a Handler (p.1852) during Logging)	1710
decaf::lang::Exception	1712
cms::ExceptionListener (If a CMS provider detects a serious problem, it notifies the client application through an ExceptionListener (p.1719) that is registered with the Connection (p.1168))	1719
activemq::commands::ExceptionResponse	1720
activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p.1722))	1722
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p.1726))	1726
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p.1730))	1730
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p.1734))	1734
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p.1738))	1738
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p.1742))	1742
decaf::util::concurrent::ExecutionException	1746
decaf::util::concurrent::Executor (An object that executes submitted decaf.lang Runnable (p.3111) tasks)	1749
decaf::util::concurrent::ExecutorService (An Executor (p.1749) that provides methods to manage termination and methods that can produce a Future (p.1840) for tracking progress of one or more asynchronous tasks)	1751
activemq::transport::failover::FailoverTransport	1752
activemq::transport::failover::FailoverTransportFactory (Creates an instance of a FailoverTransport (p.1752))	1763
activemq::transport::failover::FailoverTransportListener (Utility class used by the Transport (p.3629) to perform the work of responding to events from the active Transport (p.3629))	1766
decaf::io::FileDescriptor (This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files)	1768
decaf::util::logging::Filter (A Filter (p.1770) can be used to provide fine grain control over what is logged, beyond the control provided by log levels)	1770
decaf::io::FilterInputStream (A FilterInputStream (p.1771) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality)	1771

decaf::io::FilterOutputStream (This class is the superclass of all classes that filter output streams)	1777
decaf::lang::Float	1780
decaf::internal::nio::FloatArrayBuffer	1791
decaf::nio::FloatBuffer (This class defines four categories of operations upon float buffers:)	1800
decaf::io::Flushable (A Flushable (p. 1811) is a destination of data that can be flushed)	1811
activemq::commands::FlushCommand	1812
activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1814))	1814
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1818))	1818
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1822))	1822
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1826))	1826
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1830))	1830
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1834))	1834
decaf::util::logging::Formatter (A Formatter (p. 1838) provides support for formatting LogRecords)	1838
decaf::util::concurrent::Future< V > (A Future (p. 1840) represents the result of an asynchronous computation)	1840
activemq::transport::correlator::FutureResponse (A container that holds a response object)	1843
decaf::security::GeneralSecurityException	1845
decaf::internal::util::GenericResource< T > (A Generic Resource (p. 3072) wraps some type and will delete it when the Resource (p. 3072) itself is deleted) . .	1847
gz_header_s	1848
gz_state	1849
decaf::util::logging::Handler (A Handler (p. 1852) object takes log messages from a Logger (p. 2237) and exports them)	1852
decaf::internal::util::HexStringParser	1856
activemq::wireformat::openwire::utils::HexTable (Maps hexadecimal strings to the value of an index into the table, i.e)	1857
decaf::net::HttpRetryException	1858
activemq::util::IdGenerator	1861
decaf::lang::exceptions::IllegalArgumentException	1863
decaf::lang::exceptions::IllegalMonitorStateException	1865
cms::IllegalStateException (This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation)	1868
decaf::lang::exceptions::IllegalStateException	1869
decaf::lang::exceptions::IllegalThreadStateException	1872
activemq::transport::inactivity::InactivityMonitor	1874
decaf::lang::exceptions::IndexOutOfBoundsException	1877
decaf::net::Inet4Address	1880

decaf::net::Inet6Address	1883
decaf::net::InetAddress (Represents an IP address)	1884
decaf::net::InetSocketAddress	1891
inflate_state	1892
decaf::util::zip::Inflater (This class uncompresses data that was compressed using the <i>DEFLATE</i> algorithm (see <i>specification</i>))	1894
decaf::util::zip::InflaterInputStream (A <i>FilterInputStream</i> that decompresses data read from the wrapped <i>InputStream</i> instance)	1902
decaf::io::InputStream (A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes)	1909
decaf::io::InputStreamReader (An <i>InputStreamReader</i> (p. 1919) is a bridge from byte streams to character streams)	1919
decaf::internal::nio::IntArrayBuffer	1921
decaf::nio::IntBuffer (This class defines four categories of operations upon int buffers:)	1931
decaf::lang::Integer	1941
activemq::commands::IntegerResponse	1956
activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 1958))	1958
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 1962))	1962
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 1966))	1966
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 1970))	1970
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 1974))	1974
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 1978))	1978
internal_state	1982
activemq::transport::mock::InternalCommandListener (Listens for Commands sent from the <i>MockTransport</i> (p. 2592))	1986
decaf::lang::exceptions::InterruptedException	1987
decaf::io::InterruptedException	1990
cms::InvalidClientIdException (This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider)	1992
cms::InvalidDestinationException (This exception must be thrown when a destination either is not understood by a provider or is no longer valid)	1993
decaf::security::InvalidKeyException	1994
decaf::nio::InvalidMarkException	1997
cms::InvalidSelectorException (This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax)	1999
decaf::lang::exceptions::InvalidStateException	2000
decaf::io::IOException	2003
activemq::transport::IOTransport (Implementation of the <i>Transport</i> (p. 3629) interface that performs marshaling of commands to IO streams)	2005
decaf::lang::Iterable< E > (Implementing this interface allows an object to be cast to an <i>Iterable</i> (p. 2011) type for generic collections API calls)	2011

decaf::util::Iterator< T > (Defines an object that can be used to iterate over the elements of a collection)	2012
activemq::commands::JournalQueueAck	2014
activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 2017))	2017
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 2021))	2021
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 2025))	2025
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 2028))	2028
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 2032))	2032
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 2036))	2036
activemq::commands::JournalTopicAck	2040
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 2045))	2045
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 2049))	2049
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 2053))	2053
activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 2057))	2057
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 2061))	2061
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 2065))	2065
activemq::commands::JournalTrace	2069
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 2072))	2072
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 2076))	2076
activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 2079))	2079
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 2083))	2083

activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (Marshaling code for Open Wire Format for JournalTraceMarshaller (p. 2087))	2087
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (Marshaling code for Open Wire Format for JournalTraceMarshaller (p. 2091))	2091
activemq::commands::JournalTransaction	2095
activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 2099))	2099
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 2102))	2102
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 2106))	2106
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 2110))	2110
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 2114))	2114
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 2118))	2118
activemq::commands::KeepAliveInfo	2122
activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (Marshaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 2125))	2125
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (Marshaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 2129))	2129
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (Marshaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 2133))	2133
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (Marshaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 2137))	2137
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (Marshaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 2141))	2141
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (Marshaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 2145))	2145
decaf::security::Key (The Key (p. 2149) interface is the top-level interface for all keys)	2149
decaf::security::KeyException	2151
decaf::security::KeyManagementException	2153
activemq::commands::LastPartialCommand	2156
activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMarshaller (p. 2158))	2158
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMarshaller (p. 2162))	2162

activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMarshaller (p. 2166))	2166
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMarshaller (p. 2170))	2170
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMarshaller (p. 2174))	2174
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMarshaller (p. 2178))	2178
decaf::util::comparators::Less< E > (Simple Less (p. 2182) Comparator (p. 1127) that compares to elements to determine if the first is less than the second) . .	2182
std::less< decaf::lang::ArrayPointer< T > > (An override of the less function object so that the Pointer objects can be stored in STL Maps, etc)	2184
std::less< decaf::lang::Pointer< T > > (An override of the less function object so that the Pointer objects can be stored in STL Maps, etc)	2185
decaf::util::logging::Level (Defines a set of standard logging levels that can be used to control logging output)	2185
decaf::util::List< E > (An ordered collection (also known as a sequence))	2190
decaf::util::ListIterator< E > (An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list)	2197
activemq::commands::LocalTransactionId	2200
activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 2204))	2204
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 2208))	2208
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 2212))	2212
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 2216))	2216
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 2220))	2220
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 2224))	2224
decaf::util::concurrent::Lock (A wrapper class around a given synchronization mechanism that provides automatic release upon destruction)	2228
decaf::util::concurrent::locks::Lock (Lock (p. 2229) implementations provide more extensive locking operations than can be obtained using synchronized statements)	2229
decaf::util::concurrent::locks::LockSupport (Basic thread blocking primitives for creating locks and other synchronization classes)	2234
decaf::util::logging::Logger (A Logger (p. 2237) object is used to log messages for a specific system or application component)	2237
decaf::util::logging::LoggerHierarchy	2249
activemq::io::LoggingInputStream	2250

activemq::io::LoggingOutputStream (OutputStream filter that just logs the data being written)	2251
activemq::transport::logging::LoggingTransport (A transport filter that logs commands as they are sent/received)	2252
decaf::util::logging::LogManager (There is a single global LogManager (p. 2254) object that is used to maintain a set of shared state about Loggers and log services)	2254
decaf::util::logging::LogRecord (LogRecord (p. 2261) objects are used to pass logging requests between the logging framework and individual log Handlers) . .	2261
decaf::util::logging::LogWriter	2266
decaf::lang::Long	2267
decaf::internal::nio::LongArrayBuffer	2281
decaf::nio::LongBuffer (This class defines four categories of operations upon long long buffers:)	2291
activemq::util::LongSequenceGenerator (This class is used to generate a sequence of long long values that are incremented each time a new value is requested) .	2302
decaf::net::MalformedURLException	2303
decaf::util::Map< K, V, COMPARATOR > (Map (p. 2305) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map)	2305
cms::MapMessage (A MapMessage (p. 2318) object is used to send a set of name-value pairs)	2318
decaf::util::logging::MarkBlockLogger (Defines a class that can be used to mark the entry and exit from scoped blocks)	2327
activemq::wireformat::MarshalAware	2328
activemq::wireformat::openwire::marshal::v6::MarshallerFactory (Used to createmarshallers for a specific version of the wire protocol)	2331
activemq::wireformat::openwire::marshal::v3::MarshallerFactory (Used to createmarshallers for a specific version of the wire protocol)	2331
activemq::wireformat::openwire::marshal::v4::MarshallerFactory (Used to createmarshallers for a specific version of the wire protocol)	2332
activemq::wireformat::openwire::marshal::v5::MarshallerFactory (Used to createmarshallers for a specific version of the wire protocol)	2333
activemq::wireformat::openwire::marshal::v1::MarshallerFactory (Used to createmarshallers for a specific version of the wire protocol)	2334
activemq::wireformat::openwire::marshal::v2::MarshallerFactory (Used to createmarshallers for a specific version of the wire protocol)	2334
activemq::util::MarshallingSupport	2335
decaf::lang::Math (The class Math (p. 2339) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions)	2339
activemq::util::MemoryUsage	2354
activemq::commands::Message	2358
cms::Message (Root of all messages)	2375
activemq::commands::MessageAck	2394
activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller (Marshaling code for Open Wire Format for MessageAckMarshaller (p. 2399))	2399
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (Marshaling code for Open Wire Format for MessageAckMarshaller (p. 2403))	2403
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (Marshaling code for Open Wire Format for MessageAckMarshaller (p. 2407))	2407

activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (Marshaling code for Open Wire Format for MessageAckMarshaller (p. 2411))	2411
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (Marshaling code for Open Wire Format for MessageAckMarshaller (p. 2415))	2415
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (Marshaling code for Open Wire Format for MessageAckMarshaller (p. 2419))	2419
cms::MessageConsumer (A client uses a MessageConsumer (p. 2423) to received messages from a destination)	2423
activemq::cmsutil::MessageCreator (Creates the user-defined message to be sent by the CmsTemplate (p. 1083))	2426
activemq::commands::MessageDispatch	2427
activemq::core::MessageDispatchChannel	2431
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p. 2438))	2438
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p. 2442))	2442
activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p. 2446))	2446
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p. 2450))	2450
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p. 2454))	2454
activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p. 2458))	2458
activemq::commands::MessageDispatchNotification	2462
activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2466))	2466
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2470))	2470
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2474))	2474
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2478))	2478
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2482))	2482
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2486))	2486
cms::MessageEnumeration (Defines an object that enumerates a collection of Messages)	2490

cms::MessageEOFException (This exception must be thrown when an unexpected end of stream has been reached when a StreamMessage (p. 3415) or BytesMessage (p. 979) is being read)	2492
cms::MessageFormatException (This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type)	2493
activemq::commands::MessageId	2494
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2499))	2499
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2503))	2503
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2507))	2507
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2510))	2510
activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2514))	2514
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2518))	2518
cms::MessageListener (A MessageListener (p. 2522) object is used to receive asynchronously delivered messages)	2522
activemq::wireformat::openwire::marshal::v5::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2523))	2523
activemq::wireformat::openwire::marshal::v3::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2527))	2527
activemq::wireformat::openwire::marshal::v2::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2532))	2532
activemq::wireformat::openwire::marshal::v4::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2536))	2536
activemq::wireformat::openwire::marshal::v1::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2540))	2540
activemq::wireformat::openwire::marshal::v6::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2544))	2544
cms::MessageNotReadableException (This exception must be thrown when a CMS client attempts to read a write-only message)	2548
cms::MessageNotWriteableException (This exception must be thrown when a CMS client attempts to write to a read-only message)	2549
cms::MessageProducer (A client uses a MessageProducer (p. 2550) object to send messages to a Destination (p. 1610))	2550
activemq::wireformat::openwire::utils::MessagePropertyInterceptor (Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties)	2557
activemq::commands::MessagePull	2563
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2568))	2568
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2572))	2572
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2576))	2576

activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2580))	2580
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2584))	2584
activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2588))	2588
activemq::transport::mock::MockTransport (The MockTransport (p. 2592) defines a base level Transport (p. 3629) class that is intended to be used in place of an a regular protocol Transport (p. 3629) such as TCP)	2592
activemq::transport::mock::MockTransportFactory (Manufactures MockTransports , which are objects that read from input streams and write to output streams)	2602
decaf::util::concurrent::Mutex (Mutex (p. 2604) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java)	2604
decaf::util::concurrent::MutexHandle	2609
decaf::internal::util::concurrent::MutexImpl	2609
decaf::internal::net::Network (Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API)	2611
activemq::commands::NetworkBridgeFilter	2613
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2617))	2617
activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2620))	2620
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2624))	2624
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2628))	2628
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2632))	2632
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2636))	2636
decaf::net::NoRouteToHostException	2640
decaf::security::NoSuchAlgorithmException	2643
decaf::lang::exceptions::NoSuchElementException	2645
decaf::security::NoSuchProviderException	2648
decaf::lang::exceptions::NullPointerException	2650
decaf::lang::Number (The abstract class Number (p. 2653) is the superclass of classes Byte (p. 884), Double (p. 1672), Float (p. 1780), Integer (p. 1941), Long (p. 2267), and Short (p. 3220))	2653
decaf::lang::exceptions::NumberFormatException	2655
cms::ObjectMessage (Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object)	2658
decaf::internal::net::ssl::openssl::OpenSSLContextSpi (Provides an SSLContext that wraps the OpenSSL API)	2658

decaf::internal::net::ssl::openssl::OpenSSLParameters (Container class for parameters that are Common to OpenSSL socket classes)	2661
decaf::internal::net::ssl::openssl::OpenSSLServerSocket (SSLServerSocket based on OpenSSL library code)	2664
decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory (SSLServerSocketFactory that creates Server Sockets that use OpenSSL)	2669
decaf::internal::net::ssl::openssl::OpenSSLSocket (Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API)	2673
decaf::internal::net::ssl::openssl::OpenSSLSocketException (Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack)	2686
decaf::internal::net::ssl::openssl::OpenSSLSocketFactory (Client Socket Factory that creates SSL based client sockets using the OpenSSL library)	2689
decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream (An output stream for reading data from an OpenSSL Socket instance)	2696
decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream (OutputStream implementation used to write data to an OpenSSLSocket (p.2673) instance)	2698
activemq::wireformat::openwire::OpenWireFormat	2700
activemq::wireformat::openwire::OpenWireFormatFactory	2712
activemq::wireformat::openwire::OpenWireFormatNegotiator	2713
activemq::wireformat::openwire::OpenWireResponseBuilder	2717
decaf::io::OutputStream (Base interface for any class that wants to represent an output stream of bytes)	2718
decaf::io::OutputStreamWriter (A class for turning a character stream into a byte stream)	2726
activemq::commands::PartialCommand	2727
activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p.2731))	2731
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p.2735))	2735
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p.2739))	2739
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p.2744))	2744
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p.2748))	2748
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p.2752))	2752
decaf::lang::Pointer< T, REFCOUNTER > (Decaf's implementation of a Smart Pointer (p.2756) that is a template on a Type and is Thread (p.3520) Safe if the default Reference Counter is used)	2756
decaf::lang::PointerComparator< T, R > (This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the Pointer (p.2756) instance)	2764
activemq::cmsutil::PooledSession (A pooled session object that wraps around a delegate session)	2765

decaf::util::concurrent::PooledThread	2777
decaf::util::concurrent::PooledThreadListener (Abstract Listener Interface for users of ThreadPool (p. 3531))	2779
decaf::net::PortUnreachableException	2781
activemq::core::PrefetchPolicy (Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP)	2783
activemq::util::PrimitiveList (List of primitives)	2787
activemq::util::PrimitiveMap (Map of named primitives)	2798
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller (This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire)	2808
activemq::util::PrimitiveValueNode::PrimitiveValue (Define a union type comprised of the various types)	2814
activemq::util::PrimitiveValueConverter (Class controls the conversion of data contained in a PrimitiveValueNode (p. 2817) from one type to another)	2816
activemq::util::PrimitiveValueNode (Class that wraps around a single value of one of the many types)	2817
decaf::security::Principal (Base interface for a principal, which can represent an individual or organization)	2831
decaf::util::PriorityQueue< E > (An unbounded priority queue based on a binary heap algorithm)	2832
activemq::commands::ProducerAck	2839
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2842))	2842
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2846))	2846
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2850))	2850
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2854))	2854
activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2858))	2858
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2862))	2862
activemq::cmsutil::ProducerCallback (Callback for sending a message to a CMS destination)	2866
activemq::cmsutil::CmsTemplate::ProducerExecutor	2867
activemq::commands::ProducerId	2869
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2874))	2874
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2878))	2878
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2881))	2881

activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2885))	2885
activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2889))	2889
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2893))	2893
activemq::commands::ProducerInfo	2897
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2902))	2902
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2906))	2906
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2910))	2910
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2914))	2914
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2918))	2918
activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2922))	2922
activemq::state::ProducerState	2926
decaf::util::Properties (Java-like properties class for mapping string names to string values)	2927
decaf::util::logging::PropertiesChangeListener (Defines the interface that classes can use to listen for change events on Properties (p. 2927))	2936
decaf::net::ProtocolException	2937
decaf::security::PublicKey (A public key)	2940
decaf::io::PushbackInputStream (A PushbackInputStream (p. 2940) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte)	2940
cms::Queue (An interface encapsulating a provider-specific queue name)	2947
decaf::util::Queue< E > (A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection)	2948
cms::QueueBrowser (This class implements in interface for browsing the messages in a Queue (p. 2947) without removing them)	2951
decaf::util::Random (Random (p. 2953) Value Generator which is used to generate a stream of pseudorandom numbers)	2953
decaf::lang::Readable (A Readable (p. 2958) is a source of characters)	2958
activemq::transport::inactivity::ReadChecker (Runnable class that is used by the {)	2959
decaf::io::Reader	2960
decaf::nio::ReadOnlyBufferException	2966
decaf::util::concurrent::locks::ReadWriteLock (A ReadWriteLock (p. 2968) maintains a pair of associated locks, one for read-only operations and one for writing)	2968
activemq::cmsutil::CmsTemplate::ReceiveExecutor	2970

activemq::core::RedeliveryPolicy (Interface for a RedeliveryPolicy (p. 2972) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back)	2972
decaf::util::concurrent::locks::ReentrantLock (A reentrant mutual exclusion Lock (p. 2229) with extended capabilities)	2977
decaf::util::concurrent::RejectedExecutionException	2984
decaf::util::concurrent::RejectedExecutionHandler (A handler for tasks that cannot be executed by a ThreadPoolExecutor (p. ??))	2987
activemq::commands::RemoveInfo	2988
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2991))	2991
activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2995))	2995
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2999))	2999
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 3003))	3003
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 3007))	3007
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 3011))	3011
activemq::commands::RemoveSubscriptionInfo	3015
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 3019))	3019
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 3023))	3023
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 3027))	3027
activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 3031))	3031
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 3035))	3035
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 3039))	3039
activemq::commands::ReplayCommand	3043
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 3046))	3046
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 3050))	3050

activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 3054))	3054
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 3058))	3058
activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 3062))	3062
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 3066))	3066
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor	3070
activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor	3071
decaf::internal::util::Resource (Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown) . .	3072
decaf::internal::util::ResourceLifecycleManager	3073
activemq::cmsutil::ResourceLifecycleManager (Manages the lifecycle of a set of CMS resources)	3074
activemq::commands::Response	3076
activemq::transport::mock::ResponseBuilder (Interface for all Protocols to imple- ment that defines the behavior of the Broker in response to messages of that protocol)	3079
activemq::transport::correlator::ResponseCorrelator (This type of transport filter is responsible for correlating asynchronous responses with requests)	3081
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 3085))	3085
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 3089))	3089
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 3093))	3093
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 3098))	3098
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 3102))	3102
activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 3107))	3107
decaf::lang::Runnable (Interface for a runnable object - defines a task that can be run by a thread)	3111
decaf::lang::Runtime	3112
decaf::lang::exceptions::RuntimeException	3114
decaf::security::SecureRandom	3116
decaf::internal::security::SecureRandomImpl (Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources)	3121
decaf::security::SecureRandomSpi (Interface class used by Security Service Providers to implement a source of secure random bytes)	3124
decaf::util::concurrent::Semaphore (A counting semaphore)	3126
activemq::cmsutil::CmsTemplate::SendExecutor	3135
decaf::net::ServerSocket (This class implements server sockets)	3136
decaf::net::ServerSocketFactory (Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies)	3145
cms::Session (A Session (p. 3148) object is a single-threaded context for producing and consuming messages)	3148

activemq::cmsutil::SessionCallback (Callback for executing any number of operations on a provided CMS Session)	3160
activemq::commands::SessionId	3161
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 3165))	3165
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 3169))	3169
activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 3173))	3173
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 3176))	3176
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 3180))	3180
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 3184))	3184
activemq::commands::SessionInfo	3188
activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 3192))	3192
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 3195))	3195
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 3199))	3199
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 3203))	3203
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 3207))	3207
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 3211))	3211
activemq::cmsutil::SessionPool (A pool of CMS sessions from the same connection and with the same acknowledge mode)	3215
activemq::state::SessionState	3217
decaf::util::Set< E > (A collection that contains no duplicate elements)	3220
decaf::lang::Short	3220
decaf::internal::nio::ShortArrayBuffer	3229
decaf::nio::ShortBuffer (This class defines four categories of operations upon short buffers:)	3238
activemq::commands::ShutdownInfo	3249
activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 3252))	3252
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 3256))	3256
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 3260))	3260
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 3264))	3264
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 3268))	3268

activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 3272))	3272
decaf::security::SignatureException	3276
decaf::util::logging::SimpleFormatter (Print a brief summary of the LogRecord (p. 2261) in a human readable format)	3278
decaf::util::logging::SimpleLogger	3279
decaf::net::Socket	3281
decaf::net::SocketAddress (Base class for protocol specific Socket (p. 3281) addresses)	3297
decaf::net::SocketError (Static utility class to simplify handling of error codes for socket operations)	3298
decaf::net::SocketException (Exception for errors when manipulating sockets)	3298
decaf::net::SocketFactory (The SocketFactory (p. 3301) is used to create Socket (p. 3281) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations)	3301
decaf::internal::net::SocketFileDescriptor (File Descriptor type used internally by Decaf Socket objects)	3305
decaf::net::SocketImpl (Acts as a base class for all physical Socket (p. 3281) implementations)	3306
decaf::net::SocketImplFactory (Factory class interface for a Factory that creates SocketImpl objects)	3314
decaf::net::SocketOptions	3315
decaf::net::SocketTimeoutException	3319
decaf::net::ssl::SSLContext (Represents an implementation of the Secure Socket (p. 3281) Layer for streaming based sockets)	3321
decaf::net::ssl::SSLContextSpi (Defines the interface that should be provided by an SSLContext (p. 3321) provider)	3324
decaf::net::ssl::SSLParameters	3326
decaf::net::ssl::SSLServerSocket (Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol)	3329
decaf::net::ssl::SSLServerSocketFactory (Factory class interface that provides methods to create SSL Server Sockets)	3335
decaf::net::ssl::SSLSocket	3337
decaf::net::ssl::SSLSocketFactory (Factory class interface for a SocketFactory (p. 3301) that can create SSLSocket (p. 3337) objects)	3345
activemq::transport::tcp::SslTransport (Transport (p. 3629) for connecting to a Broker using an SSL Socket)	3347
activemq::transport::tcp::SslTransportFactory	3349
activemq::commands::BrokerError::StackTraceElement	3350
decaf::internal::io::StandardErrorOutputStream (Wrapper Around the Standard error Output facility on the current platform)	3351
decaf::internal::io::StandardInputStream	3352
decaf::internal::io::StandardOutputStream	3354
cms::Startable (Interface for a class that implements the start method)	3355
decaf::lang::STATIC_CAST_TOKEN	3356
activemq::core::ActiveMQConstants::StaticInitializer	3356
decaf::util::StlList< E > (List (p. 2190) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type)	3357
decaf::util::StlMap< K, V, COMPARATOR > (Map (p. 2305) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map)	3369
decaf::util::StlQueue< T > (The Queue (p. 2948) class accepts messages with an psuh(m) command where m is the message to be queued)	3382

decaf::util::StlSet< E > (Set (p. 3220) template that wraps around a <code>std::set</code> to provide a more user-friendly interface and to provide common functions that do not exist in <code>std::set</code>)	3390
activemq::wireformat::stomp::StompCommandConstants	3395
activemq::wireformat::stomp::StompFrame (A Stomp-level message frame that encloses all messages to and from the broker)	3398
activemq::wireformat::stomp::StompHelper (Utility Methods used when marshaling to and from <code>StompFrame</code> 's)	3402
activemq::wireformat::stomp::StompWireFormat	3407
activemq::wireformat::stomp::StompWireFormatFactory (Factory used to create the <code>Stomp Wire Format</code> instance)	3410
cms::Stoppable (Interface for a class that implements the stop method)	3411
decaf::util::logging::StreamHandler (Stream based logging Handler (p. 1852))	3412
cms::StreamMessage (Interface for a StreamMessage (p. 3415))	3415
decaf::lang::String (Immutable sequence of chars)	3427
decaf::util::StringTokenizer	3430
activemq::commands::SubscriptionInfo	3433
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 3438))	3438
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 3442))	3442
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 3446))	3446
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 3450))	3450
activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 3453))	3453
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 3457))	3457
decaf::util::concurrent::Synchronizable (The interface for all synchronizable objects (that is, objects that can be locked and unlocked))	3461
decaf::internal::util::concurrent::SynchronizableImpl (A convenience class used by some Decaf classes to implement the <code>Synchronizable</code> interface when there is no issues related to multiple inheritance)	3473
activemq::core::Synchronization (Transacted Object Synchronization (p. 3477), used to sync the events of a Transaction with the items in the Transaction)	3477
decaf::util::concurrent::SynchronousQueue< E > (A blocking queue (p. 774) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa)	3477
decaf::lang::System (Static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays)	3487
activemq::threads::Task (Represents a unit of work that requires one or more iterations to complete)	3494
decaf::util::concurrent::TaskListener	3495
activemq::threads::TaskRunner	3496

decaf::internal::net::tcp::TcpSocket (Platform-independent implementation of the socket interface)	3497
decaf::internal::net::tcp::TcpSocketInputStream (Input stream for performing reads on a socket)	3506
decaf::internal::net::tcp::TcpSocketOutputStream (Output stream for performing write operations on a socket)	3509
activemq::transport::tcp::TcpTransport (Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an IOTransport (p. 2005))	3510
activemq::transport::tcp::TcpTransportFactory (Factory Responsible for creating the TcpTransport (p. 3510))	3514
cms::TemporaryQueue (Defines a Temporary Queue (p. 2947) based Destination (p. 1610))	3516
cms::TemporaryTopic (Defines a Temporary Topic (p. 3568) based Destination (p. 1610))	3517
cms::TextMessage (Interface for a text message)	3519
decaf::lang::Thread (A Thread (p. 3520) is a concurrent unit of execution)	3520
decaf::util::concurrent::ThreadFactory (Public interface ThreadFactory (p. 3529))	3529
decaf::lang::ThreadGroup	3531
decaf::util::concurrent::ThreadPool (Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks)	3531
decaf::lang::Throwable (This class represents an error that has occurred)	3536
decaf::util::concurrent::TimeoutException	3541
decaf::util::Timer (A facility for threads to schedule tasks for future execution in a background thread)	3543
decaf::util::TimerTask (A Base class for a task object that can be scheduled for one-time or repeated execution by a Timer (p. 3543))	3554
decaf::internal::util::TimerTaskHeap (A Binary Heap implemented specifically for the Timer class in Decaf Util)	3556
decaf::util::concurrent::TimeUnit (A TimeUnit (p. 3559) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units)	3559
cms::Topic (An interface encapsulating a provider-specific topic name)	3568
activemq::state::Tracked	3569
activemq::commands::TransactionId	3569
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3572))	3572
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3576))	3576
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3580))	3580
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3583))	3583
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3587))	3587
activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3591))	3591
activemq::commands::TransactionInfo	3594

activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3598))	3598
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3602))	3602
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3606))	3606
activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3610))	3610
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3614))	3614
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3618))	3618
activemq::state::TransactionState	3622
decaf::internal::util::concurrent::Transferer< E > (Shared internal API for dual stacks and queues)	3625
decaf::internal::util::concurrent::TransferQueue< E > (This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers)	3625
decaf::internal::util::concurrent::TransferStack< E >	3627
activemq::transport::Transport (Interface for a transport layer for command objects)	3629
activemq::transport::TransportFactory (Defines the interface for Factories that cre- ate Transports or TransportFilters)	3634
activemq::transport::TransportFilter (A filter on the transport layer)	3636
activemq::transport::TransportListener (A listener of asynchronous exceptions from a command transport object)	3643
activemq::transport::TransportRegistry (Registry of all Transport (p. 3629) Fac- tories that are available to the client at runtime)	3645
tree_desc_s	3648
decaf::lang::Thread::UncaughtExceptionHandler (Interface for handlers invoked when a Thread (p. 3520) abruptly terminates due to an uncaught exception) .	3648
decaf::net::UnknownHostException	3649
decaf::net::UnknownServiceException	3652
decaf::io::UnsupportedEncodingException (Thrown when the the Character En- coding is not supported)	3654
decaf::lang::exceptions::UnsupportedOperationException	3657
cms::UnsupportedOperationException (This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not sup- ported by the CMS Provider in use)	3659
decaf::net::URI (This class represents an instance of a URI (p. 3660) as defined by RFC 2396)	3660
decaf::internal::net::URIEncoderDecoder	3672
decaf::internal::net::URIHelper (Helper class used by the URI classes in encoding and decoding of URI's)	3674
activemq::transport::failover::URIPool	3681
activemq::util::URISupport	3684
decaf::net::URISyntaxException	3686
decaf::internal::net::URIType (Basic type object that holds data that composes a given URI)	3690

decaf::net::URL (Class URL (p. 3697) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web)	3697
decaf::net::URLDecoder	3699
decaf::net::URLEncoder	3700
activemq::util::Usage	3701
decaf::io::UTFDataFormatException (Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered)	3703
decaf::util::UUID (A class that represents an immutable universally unique identifier (UUID (p. 3706)))	3706
activemq::wireformat::WireFormat (Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams) .	3712
activemq::wireformat::WireFormatFactory (The WireFormatFactory (p. 3716) is the interface that all WireFormatFactory (p. 3716) classes must extend) .	3716
activemq::commands::WireFormatInfo	3717
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p. 3728))	3728
activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p. 3731))	3731
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p. 3735))	3735
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p. 3739))	3739
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p. 3743))	3743
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p. 3747))	3747
activemq::wireformat::WireFormatNegotiator (Defines a WireFormatNegotiator (p. 3751) which allows a WireFormat (p. 3712) to)	3751
activemq::wireformat::WireFormatRegistry (Registry of all WireFormat (p. 3712) Factories that are available to the client at runtime)	3752
activemq::transport::inactivity::WriteChecker (Runnable class used by the { } . .	3755
decaf::io::Writer	3756
decaf::security::auth::x500::X500Principal	3761
decaf::security::cert::X509Certificate (Base interface for all identity certificates) .	3762
activemq::commands::XATransactionId	3765
activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 3769))	3769
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 3773))	3773
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 3777))	3777
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 3781))	3781

activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 3785))	3785
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 3789))	3789
decaf::util::logging::XMLFormatter (Format a LogRecord (p. 2261) into a standard XML format)	3793
z_stream_s	3795
decaf::util::zip::ZipException	3796

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/main/activemq/cmsutil/	CachedConsumer.h	3799
src/main/activemq/cmsutil/	CachedProducer.h	3799
src/main/activemq/cmsutil/	CmsAccessor.h	3800
src/main/activemq/cmsutil/	CmsDestinationAccessor.h	3800
src/main/activemq/cmsutil/	CmsTemplate.h	3801
src/main/activemq/cmsutil/	DestinationResolver.h	3801
src/main/activemq/cmsutil/	DynamicDestinationResolver.h	3802
src/main/activemq/cmsutil/	MessageCreator.h	3802
src/main/activemq/cmsutil/	PooledSession.h	3803
src/main/activemq/cmsutil/	ProducerCallback.h	3803
src/main/activemq/cmsutil/	ResourceLifecycleManager.h	3804
src/main/activemq/cmsutil/	SessionCallback.h	3805
src/main/activemq/cmsutil/	SessionPool.h	3805
src/main/activemq/commands/	ActiveMQBlobMessage.h	3806
src/main/activemq/commands/	ActiveMQBytesMessage.h	3806
src/main/activemq/commands/	ActiveMQDestination.h	3807
src/main/activemq/commands/	ActiveMQMapMessage.h	3808
src/main/activemq/commands/	ActiveMQMessage.h	3808
src/main/activemq/commands/	ActiveMQMessageTemplate.h	3809
src/main/activemq/commands/	ActiveMQObjectMessage.h	3809
src/main/activemq/commands/	ActiveMQQueue.h	3810
src/main/activemq/commands/	ActiveMQStreamMessage.h	3810
src/main/activemq/commands/	ActiveMQTempDestination.h	3811
src/main/activemq/commands/	ActiveMQTempQueue.h	3812
src/main/activemq/commands/	ActiveMQTempTopic.h	3812
src/main/activemq/commands/	ActiveMQTextMessage.h	3813
src/main/activemq/commands/	ActiveMQTopic.h	3813
src/main/activemq/commands/	BaseCommand.h	3814
src/main/activemq/commands/	BaseDataStructure.h	3814
src/main/activemq/commands/	BooleanExpression.h	3815
src/main/activemq/commands/	BrokerError.h	3815
src/main/activemq/commands/	BrokerId.h	3816
src/main/activemq/commands/	BrokerInfo.h	3816

src/main/activemq/commands/	Command.h	3817
src/main/activemq/commands/	ConnectionControl.h	3817
src/main/activemq/commands/	ConnectionError.h	3818
src/main/activemq/commands/	ConnectionId.h	3818
src/main/activemq/commands/	ConnectionInfo.h	3819
src/main/activemq/commands/	ConsumerControl.h	3819
src/main/activemq/commands/	ConsumerId.h	3820
src/main/activemq/commands/	ConsumerInfo.h	3821
src/main/activemq/commands/	ControlCommand.h	3821
src/main/activemq/commands/	DataArrayResponse.h	3822
src/main/activemq/commands/	DataResponse.h	3822
src/main/activemq/commands/	DataStructure.h	3823
src/main/activemq/commands/	DestinationInfo.h	3823
src/main/activemq/commands/	DiscoveryEvent.h	3824
src/main/activemq/commands/	ExceptionResponse.h	3824
src/main/activemq/commands/	FlushCommand.h	3825
src/main/activemq/commands/	IntegerResponse.h	3825
src/main/activemq/commands/	JournalQueueAck.h	3826
src/main/activemq/commands/	JournalTopicAck.h	3826
src/main/activemq/commands/	JournalTrace.h	3827
src/main/activemq/commands/	JournalTransaction.h	3827
src/main/activemq/commands/	KeepAliveInfo.h	3828
src/main/activemq/commands/	LastPartialCommand.h	3828
src/main/activemq/commands/	LocalTransactionId.h	3829
src/main/activemq/commands/	Message.h	3829
src/main/activemq/commands/	MessageAck.h	3831
src/main/activemq/commands/	MessageDispatch.h	3831
src/main/activemq/commands/	MessageDispatchNotification.h	3832
src/main/activemq/commands/	MessageId.h	3832
src/main/activemq/commands/	MessagePull.h	3833
src/main/activemq/commands/	NetworkBridgeFilter.h	3834
src/main/activemq/commands/	PartialCommand.h	3834
src/main/activemq/commands/	ProducerAck.h	3835
src/main/activemq/commands/	ProducerId.h	3835
src/main/activemq/commands/	ProducerInfo.h	3836
src/main/activemq/commands/	RemoveInfo.h	3836
src/main/activemq/commands/	RemoveSubscriptionInfo.h	3837
src/main/activemq/commands/	ReplayCommand.h	3837
src/main/activemq/commands/	Response.h	3838
src/main/activemq/commands/	SessionId.h	3838
src/main/activemq/commands/	SessionInfo.h	3839
src/main/activemq/commands/	ShutdownInfo.h	3839
src/main/activemq/commands/	SubscriptionInfo.h	3840
src/main/activemq/commands/	TransactionId.h	3840
src/main/activemq/commands/	TransactionInfo.h	3841
src/main/activemq/commands/	WireFormatInfo.h	3841
src/main/activemq/commands/	XATransactionId.h	3842
src/main/activemq/core/	ActiveMQAckHandler.h	3842
src/main/activemq/core/	ActiveMQConnection.h	3843
src/main/activemq/core/	ActiveMQConnectionFactory.h	3844
src/main/activemq/core/	ActiveMQConnectionMetaData.h	3844
src/main/activemq/core/	ActiveMQConstants.h	3845
src/main/activemq/core/	ActiveMQConsumer.h	3845
src/main/activemq/core/	ActiveMQProducer.h	3846

src/main/activemq/core/ActiveMQQueueBrowser.h	3847
src/main/activemq/core/ActiveMQSession.h	3847
src/main/activemq/core/ActiveMQSessionExecutor.h	3848
src/main/activemq/core/ActiveMQTransactionContext.h	3849
src/main/activemq/core/DispatchData.h	3849
src/main/activemq/core/Dispatcher.h	3850
src/main/activemq/core/MessageDispatchChannel.h	3850
src/main/activemq/core/PrefetchPolicy.h	3852
src/main/activemq/core/RedeliveryPolicy.h	3852
src/main/activemq/core/Synchronization.h	3853
src/main/activemq/core/policies/DefaultPrefetchPolicy.h	3851
src/main/activemq/core/policies/DefaultRedeliveryPolicy.h	3851
src/main/activemq/exceptions/ActiveMQException.h	3853
src/main/activemq/exceptions/BrokerException.h	3854
src/main/activemq/exceptions/ExceptionDefines.h	3854
src/main/activemq/io/LoggingInputStream.h	3858
src/main/activemq/io/LoggingOutputStream.h	3859
src/main/activemq/library/ActiveMQCPP.h	3859
src/main/activemq/state/CommandVisitor.h	3860
src/main/activemq/state/CommandVisitorAdapter.h	3860
src/main/activemq/state/ConnectionState.h	3861
src/main/activemq/state/ConnectionStateTracker.h	3862
src/main/activemq/state/ConsumerState.h	3863
src/main/activemq/state/ProducerState.h	3863
src/main/activemq/state/SessionState.h	3864
src/main/activemq/state/Tracked.h	3865
src/main/activemq/state/TransactionState.h	3865
src/main/activemq/threads/CompositeTask.h	3866
src/main/activemq/threads/CompositeTaskRunner.h	3866
src/main/activemq/threads/DedicatedTaskRunner.h	3867
src/main/activemq/threads/Task.h	3867
src/main/activemq/threads/TaskRunner.h	3868
src/main/activemq/transport/AbstractTransportFactory.h	3868
src/main/activemq/transport/CompositeTransport.h	3869
src/main/activemq/transport/DefaultTransportListener.h	3871
src/main/activemq/transport/IOTransport.h	3877
src/main/activemq/transport/Transport.h	3883
src/main/activemq/transport/TransportFactory.h	3884
src/main/activemq/transport/TransportFilter.h	3884
src/main/activemq/transport/TransportListener.h	3885
src/main/activemq/transport/TransportRegistry.h	3885
src/main/activemq/transport/correlator/FutureResponse.h	3869
src/main/activemq/transport/correlator/ResponseCorrelator.h	3870
src/main/activemq/transport/failover/BackupTransport.h	3871
src/main/activemq/transport/failover/BackupTransportPool.h	3872
src/main/activemq/transport/failover/CloseTransportsTask.h	3872
src/main/activemq/transport/failover/FailoverTransport.h	3873
src/main/activemq/transport/failover/FailoverTransportFactory.h	3874
src/main/activemq/transport/failover/FailoverTransportListener.h	3874
src/main/activemq/transport/failover/URIPool.h	3875
src/main/activemq/transport/inactivity/InactivityMonitor.h	3875
src/main/activemq/transport/inactivity/ReadChecker.h	3876
src/main/activemq/transport/inactivity/WriteChecker.h	3876
src/main/activemq/transport/logging/LoggingTransport.h	3878

src/main/activemq/transport/mock/ InternalCommandListener.h	3878
src/main/activemq/transport/mock/ MockTransport.h	3879
src/main/activemq/transport/mock/ MockTransportFactory.h	3880
src/main/activemq/transport/mock/ ResponseBuilder.h	3880
src/main/activemq/transport/tcp/ SslTransport.h	3881
src/main/activemq/transport/tcp/ SslTransportFactory.h	3881
src/main/activemq/transport/tcp/ TcpTransport.h	3882
src/main/activemq/transport/tcp/ TcpTransportFactory.h	3882
src/main/activemq/util/ ActiveMQProperties.h	3886
src/main/activemq/util/ CMSExceptionSupport.h	3886
src/main/activemq/util/ CompositeData.h	3888
src/main/activemq/util/ Config.h	3889
src/main/activemq/util/ IdGenerator.h	3890
src/main/activemq/util/ LongSequenceGenerator.h	3890
src/main/activemq/util/ MarshallingSupport.h	3891
src/main/activemq/util/ MemoryUsage.h	3891
src/main/activemq/util/ PrimitiveList.h	3892
src/main/activemq/util/ PrimitiveMap.h	3892
src/main/activemq/util/ PrimitiveValueConverter.h	3893
src/main/activemq/util/ PrimitiveValueNode.h	3893
src/main/activemq/util/ URISupport.h	3894
src/main/activemq/util/ Usage.h	3894
src/main/activemq/wireformat/ MarshalAware.h	3895
src/main/activemq/wireformat/ WireFormat.h	4151
src/main/activemq/wireformat/ WireFormatFactory.h	4152
src/main/activemq/wireformat/ WireFormatNegotiator.h	4153
src/main/activemq/wireformat/ WireFormatRegistry.h	4153
src/main/activemq/wireformat/openwire/ OpenWireFormat.h	4144
src/main/activemq/wireformat/openwire/ OpenWireFormatFactory.h	4145
src/main/activemq/wireformat/openwire/ OpenWireFormatNegotiator.h	4146
src/main/activemq/wireformat/openwire/ OpenWireResponseBuilder.h	4146
src/main/activemq/wireformat/openwire/marshal/ BaseDataStreamMarshaller.h	3895
src/main/activemq/wireformat/openwire/marshal/ DataStreamMarshaller.h	3896
src/main/activemq/wireformat/openwire/marshal/ PrimitiveTypesMarshaller.h	3897
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQBlobMessageMarshaller.h	3897
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQBytesMessageMarshaller.h	3901
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQDestinationMarshaller.h	3905
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQMapMessageMarshaller.h	3909
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQMessageMarshaller.h	3913
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQObjectMessageMarshaller.h	3917
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQQueueMarshaller.h	3921
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQStreamMessageMarshaller.h	3925
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTempDestinationMarshaller.h	3929
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTempQueueMarshaller.h	3933

src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h	
3937	
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h	
3941	
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h	3945
src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h	3949
src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h . . .	3953
src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h . . .	3957
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h	
3961	
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h	
3965	
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h .	3969
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h	3973
src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h	
3977	
src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h . .	3981
src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h .	3985
src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h	
3989	
src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h	
3993	
src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h .	3997
src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h	4001
src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h	4005
src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h	
4009	
src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h	4013
src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h	
4017	
src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h	
4021	
src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h	
4025	
src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h . .	4029
src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h	
4033	
src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h .	4037
src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h	
4041	
src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h	
4045	
src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h	4049
src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h . .	4052
src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h	
4056	
src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h	
4060	
src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h . . .	4064
src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h	4068
src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h . .	4072
src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h	
4076	

src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h	
4080	
src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h . . .	4084
src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h . . .	4088
src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h . . .	4092
src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h . . .	4096
src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h	
4100	
src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h	
4104	
src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h	4108
src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h	4112
src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h . . .	4116
src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h .	4120
src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h	
4124	
src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h .	4128
src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h	4132
src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h	4136
src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h	
4140	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h	
3898	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h	
3902	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h	
3906	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h	
3910	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h	
3914	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h	
3918	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h	
3922	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h	
3926	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h	
3930	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h	
3934	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h	
3938	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h	
3942	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h	3946
src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h	3950
src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h	3954
src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h . . .	3958
src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h	
3962	
src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h	
3966	
src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h .	3970

src/main/activemq/wireformat/openwire/marshall/v2/	ConnectionInfoMarshaller.h	3974
src/main/activemq/wireformat/openwire/marshall/v2/	ConsumerControlMarshaller.h	3978
src/main/activemq/wireformat/openwire/marshall/v2/	ConsumerIdMarshaller.h	3982
src/main/activemq/wireformat/openwire/marshall/v2/	ConsumerInfoMarshaller.h	3986
src/main/activemq/wireformat/openwire/marshall/v2/	ControlCommandMarshaller.h	3990
src/main/activemq/wireformat/openwire/marshall/v2/	DataArrayResponseMarshaller.h	3994
src/main/activemq/wireformat/openwire/marshall/v2/	DataResponseMarshaller.h	3998
src/main/activemq/wireformat/openwire/marshall/v2/	DestinationInfoMarshaller.h	4002
src/main/activemq/wireformat/openwire/marshall/v2/	DiscoveryEventMarshaller.h	4006
src/main/activemq/wireformat/openwire/marshall/v2/	ExceptionResponseMarshaller.h	4010
src/main/activemq/wireformat/openwire/marshall/v2/	FlushCommandMarshaller.h	4014
src/main/activemq/wireformat/openwire/marshall/v2/	IntegerResponseMarshaller.h	4018
src/main/activemq/wireformat/openwire/marshall/v2/	JournalQueueAckMarshaller.h	4022
src/main/activemq/wireformat/openwire/marshall/v2/	JournalTopicAckMarshaller.h	4026
src/main/activemq/wireformat/openwire/marshall/v2/	JournalTraceMarshaller.h	4030
src/main/activemq/wireformat/openwire/marshall/v2/	JournalTransactionMarshaller.h	4034
src/main/activemq/wireformat/openwire/marshall/v2/	KeepAliveInfoMarshaller.h	4038
src/main/activemq/wireformat/openwire/marshall/v2/	LastPartialCommandMarshaller.h	4042
src/main/activemq/wireformat/openwire/marshall/v2/	LocalTransactionIdMarshaller.h	4046
src/main/activemq/wireformat/openwire/marshall/v2/	MarshallerFactory.h	4050
src/main/activemq/wireformat/openwire/marshall/v2/	MessageAckMarshaller.h	4053
src/main/activemq/wireformat/openwire/marshall/v2/	MessageDispatchMarshaller.h	4057
src/main/activemq/wireformat/openwire/marshall/v2/	MessageDispatchNotificationMarshaller.h	4061
src/main/activemq/wireformat/openwire/marshall/v2/	MessageIdMarshaller.h	4065
src/main/activemq/wireformat/openwire/marshall/v2/	MessageMarshaller.h	4069
src/main/activemq/wireformat/openwire/marshall/v2/	MessagePullMarshaller.h	4073
src/main/activemq/wireformat/openwire/marshall/v2/	NetworkBridgeFilterMarshaller.h	4077
src/main/activemq/wireformat/openwire/marshall/v2/	PartialCommandMarshaller.h	4081
src/main/activemq/wireformat/openwire/marshall/v2/	ProducerAckMarshaller.h	4085
src/main/activemq/wireformat/openwire/marshall/v2/	ProducerIdMarshaller.h	4089
src/main/activemq/wireformat/openwire/marshall/v2/	ProducerInfoMarshaller.h	4093
src/main/activemq/wireformat/openwire/marshall/v2/	RemoveInfoMarshaller.h	4097
src/main/activemq/wireformat/openwire/marshall/v2/	RemoveSubscriptionInfoMarshaller.h	4101
src/main/activemq/wireformat/openwire/marshall/v2/	ReplayCommandMarshaller.h	4105
src/main/activemq/wireformat/openwire/marshall/v2/	ResponseMarshaller.h	4109
src/main/activemq/wireformat/openwire/marshall/v2/	SessionIdMarshaller.h	4113
src/main/activemq/wireformat/openwire/marshall/v2/	SessionInfoMarshaller.h	4117
src/main/activemq/wireformat/openwire/marshall/v2/	ShutdownInfoMarshaller.h	4121

src/main/activemq/wireformat/openwire/marshal/v2/**SubscriptionInfoMarshaller.h**
 4125
 src/main/activemq/wireformat/openwire/marshal/v2/**TransactionIdMarshaller.h** . 4129
 src/main/activemq/wireformat/openwire/marshal/v2/**TransactionInfoMarshaller.h** 4133
 src/main/activemq/wireformat/openwire/marshal/v2/**WireFormatInfoMarshaller.h** 4137
 src/main/activemq/wireformat/openwire/marshal/v2/**XATransactionIdMarshaller.h**
 4141
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQBlobMessageMarshaller.h**
 3899
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQBytesMessageMarshaller.h**
 3903
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQDestinationMarshaller.h**
 3907
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMapMessageMarshaller.h**
 3911
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMessageMarshaller.h**
 3915
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQObjectMessageMarshaller.h**
 3919
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQQueueMarshaller.h**
 3923
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQStreamMessageMarshaller.h**
 3927
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempDestinationMarshaller.h**
 3931
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempQueueMarshaller.h**
 3935
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempTopicMarshaller.h**
 3939
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTextMessageMarshaller.h**
 3943
 src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTopicMarshaller.h** 3947
 src/main/activemq/wireformat/openwire/marshal/v3/**BaseCommandMarshaller.h** 3951
 src/main/activemq/wireformat/openwire/marshal/v3/**BrokerIdMarshaller.h** . . . 3955
 src/main/activemq/wireformat/openwire/marshal/v3/**BrokerInfoMarshaller.h** . . . 3959
 src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionControlMarshaller.h**
 3963
 src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionErrorMarshaller.h**
 3967
 src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionIdMarshaller.h** . 3971
 src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionInfoMarshaller.h** 3975
 src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerControlMarshaller.h**
 3979
 src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerIdMarshaller.h** . . 3983
 src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerInfoMarshaller.h** . 3987
 src/main/activemq/wireformat/openwire/marshal/v3/**ControlCommandMarshaller.h**
 3991
 src/main/activemq/wireformat/openwire/marshal/v3/**DataArrayResponseMarshaller.h**
 3995
 src/main/activemq/wireformat/openwire/marshal/v3/**DataResponseMarshaller.h** . 3999
 src/main/activemq/wireformat/openwire/marshal/v3/**DestinationInfoMarshaller.h** 4003
 src/main/activemq/wireformat/openwire/marshal/v3/**DiscoveryEventMarshaller.h** 4007
 src/main/activemq/wireformat/openwire/marshal/v3/**ExceptionResponseMarshaller.h**
 4011

src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h	4015
src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h	4019
src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h	4023
src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h	4027
src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h	4031
src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h	4035
src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h	4039
src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h	4043
src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h	4047
src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h	4050
src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h	4053
src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h	4057
src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h	4062
src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h	4066
src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h	4070
src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h	4074
src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h	4078
src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h	4082
src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h	4086
src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h	4090
src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h	4094
src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h	4098
src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h	4102
src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h	4106
src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h	4110
src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h	4114
src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h	4118
src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h	4122
src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h	4126
src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h	4130
src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h	4134
src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h	4138
src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h	4142
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h	3899
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h	3903
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h	3907

src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQMapMessageMarshaller.h	
3911	
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQMessageMarshaller.h	
3915	
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQObjectMessageMarshaller.h	
3919	
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQQueueMarshaller.h	
3923	
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQStreamMessageMarshaller.h	
3927	
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQTempDestinationMarshaller.h	
3931	
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQTempQueueMarshaller.h	
3935	
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQTempTopicMarshaller.h	
3939	
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQTextMessageMarshaller.h	
3943	
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQTopicMarshaller.h	3947
src/main/activemq/wireformat/openwire/marshal/v4/ BaseCommandMarshaller.h	3951
src/main/activemq/wireformat/openwire/marshal/v4/ BrokerIdMarshaller.h	3955
src/main/activemq/wireformat/openwire/marshal/v4/ BrokerInfoMarshaller.h	3959
src/main/activemq/wireformat/openwire/marshal/v4/ ConnectionControlMarshaller.h	
3963	
src/main/activemq/wireformat/openwire/marshal/v4/ ConnectionErrorMarshaller.h	
3967	
src/main/activemq/wireformat/openwire/marshal/v4/ ConnectionIdMarshaller.h	3971
src/main/activemq/wireformat/openwire/marshal/v4/ ConnectionInfoMarshaller.h	3975
src/main/activemq/wireformat/openwire/marshal/v4/ ConsumerControlMarshaller.h	
3979	
src/main/activemq/wireformat/openwire/marshal/v4/ ConsumerIdMarshaller.h	3983
src/main/activemq/wireformat/openwire/marshal/v4/ ConsumerInfoMarshaller.h	3987
src/main/activemq/wireformat/openwire/marshal/v4/ ControlCommandMarshaller.h	
3991	
src/main/activemq/wireformat/openwire/marshal/v4/ DataArrayResponseMarshaller.h	
3995	
src/main/activemq/wireformat/openwire/marshal/v4/ DataResponseMarshaller.h	3999
src/main/activemq/wireformat/openwire/marshal/v4/ DestinationInfoMarshaller.h	4003
src/main/activemq/wireformat/openwire/marshal/v4/ DiscoveryEventMarshaller.h	4007
src/main/activemq/wireformat/openwire/marshal/v4/ ExceptionResponseMarshaller.h	
4011	
src/main/activemq/wireformat/openwire/marshal/v4/ FlushCommandMarshaller.h	4015
src/main/activemq/wireformat/openwire/marshal/v4/ IntegerResponseMarshaller.h	
4019	
src/main/activemq/wireformat/openwire/marshal/v4/ JournalQueueAckMarshaller.h	
4023	
src/main/activemq/wireformat/openwire/marshal/v4/ JournalTopicAckMarshaller.h	
4027	
src/main/activemq/wireformat/openwire/marshal/v4/ JournalTraceMarshaller.h	4031
src/main/activemq/wireformat/openwire/marshal/v4/ JournalTransactionMarshaller.h	
4035	
src/main/activemq/wireformat/openwire/marshal/v4/ KeepAliveInfoMarshaller.h	4039
src/main/activemq/wireformat/openwire/marshal/v4/ LastPartialCommandMarshaller.h	
4043	

src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h	
4047	
src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h	4051
src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h . .	4054
src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h	
4058	
src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h	
4062	
src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h . . .	4066
src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h	4070
src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h . .	4074
src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h	
4078	
src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h	
4082	
src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h . .	4086
src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h . . .	4090
src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h . .	4094
src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h . .	4098
src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h	
4102	
src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h	
4106	
src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h	4110
src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h	4114
src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h . . .	4118
src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h .	4122
src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h	
4126	
src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h .	4130
src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h	4134
src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h	4138
src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h	
4142	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h	
3900	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h	
3904	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h	
3908	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h	
3912	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h	
3916	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h	
3920	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h	
3924	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h	
3928	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h	
3932	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h	
3936	

src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h	
3940	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h	
3944	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h	3948
src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h	3952
src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h . . .	3956
src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h . . .	3960
src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h	
3964	
src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h	
3968	
src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h .	3972
src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h	3976
src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h	
3980	
src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h . .	3984
src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h .	3988
src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h	
3992	
src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h	
3996	
src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h .	4000
src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h	4004
src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h	4008
src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h	
4012	
src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h	4016
src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h	
4020	
src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h	
4024	
src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h	
4028	
src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h . .	4032
src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h	
4036	
src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h .	4040
src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h	
4044	
src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h	
4048	
src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h	4051
src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h . .	4055
src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h	
4059	
src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h	
4063	
src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h . . .	4067
src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h	4071
src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h . .	4075
src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h	
4079	

src/main/activemq/wireformat/openwire/marshall/v5/PartialCommandMarshaller.h	
4083	
src/main/activemq/wireformat/openwire/marshall/v5/ProducerAckMarshaller.h	4087
src/main/activemq/wireformat/openwire/marshall/v5/ProducerIdMarshaller.h	4091
src/main/activemq/wireformat/openwire/marshall/v5/ProducerInfoMarshaller.h	4095
src/main/activemq/wireformat/openwire/marshall/v5/RemoveInfoMarshaller.h	4099
src/main/activemq/wireformat/openwire/marshall/v5/RemoveSubscriptionInfoMarshaller.h	
4103	
src/main/activemq/wireformat/openwire/marshall/v5/ReplayCommandMarshaller.h	
4107	
src/main/activemq/wireformat/openwire/marshall/v5/ResponseMarshaller.h	4111
src/main/activemq/wireformat/openwire/marshall/v5/SessionIdMarshaller.h	4115
src/main/activemq/wireformat/openwire/marshall/v5/SessionInfoMarshaller.h	4119
src/main/activemq/wireformat/openwire/marshall/v5/ShutdownInfoMarshaller.h	4123
src/main/activemq/wireformat/openwire/marshall/v5/SubscriptionInfoMarshaller.h	
4127	
src/main/activemq/wireformat/openwire/marshall/v5/TransactionIdMarshaller.h	4131
src/main/activemq/wireformat/openwire/marshall/v5/TransactionInfoMarshaller.h	4135
src/main/activemq/wireformat/openwire/marshall/v5/WireFormatInfoMarshaller.h	4139
src/main/activemq/wireformat/openwire/marshall/v5/XATransactionIdMarshaller.h	
4143	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQBlobMessageMarshaller.h	
3901	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQBytesMessageMarshaller.h	
3905	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQDestinationMarshaller.h	
3909	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQMapMessageMarshaller.h	
3913	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQMessageMarshaller.h	
3917	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQObjectMessageMarshaller.h	
3921	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQQueueMarshaller.h	
3925	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQStreamMessageMarshaller.h	
3929	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTempDestinationMarshaller.h	
3933	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTempQueueMarshaller.h	
3937	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTempTopicMarshaller.h	
3941	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTextMessageMarshaller.h	
3945	
src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTopicMarshaller.h	3949
src/main/activemq/wireformat/openwire/marshall/v6/BaseCommandMarshaller.h	3953
src/main/activemq/wireformat/openwire/marshall/v6/BrokerIdMarshaller.h	3957
src/main/activemq/wireformat/openwire/marshall/v6/BrokerInfoMarshaller.h	3961
src/main/activemq/wireformat/openwire/marshall/v6/ConnectionControlMarshaller.h	
3965	
src/main/activemq/wireformat/openwire/marshall/v6/ConnectionErrorMarshaller.h	
3969	
src/main/activemq/wireformat/openwire/marshall/v6/ConnectionIdMarshaller.h	3973

src/main/activemq/wireformat/openwire/marshal/v6/	ConnectionInfoMarshaller.h	3977
src/main/activemq/wireformat/openwire/marshal/v6/	ConsumerControlMarshaller.h	3981
src/main/activemq/wireformat/openwire/marshal/v6/	ConsumerIdMarshaller.h	3985
src/main/activemq/wireformat/openwire/marshal/v6/	ConsumerInfoMarshaller.h	3989
src/main/activemq/wireformat/openwire/marshal/v6/	ControlCommandMarshaller.h	3993
src/main/activemq/wireformat/openwire/marshal/v6/	DataArrayResponseMarshaller.h	3997
src/main/activemq/wireformat/openwire/marshal/v6/	DataResponseMarshaller.h	4001
src/main/activemq/wireformat/openwire/marshal/v6/	DestinationInfoMarshaller.h	4005
src/main/activemq/wireformat/openwire/marshal/v6/	DiscoveryEventMarshaller.h	4009
src/main/activemq/wireformat/openwire/marshal/v6/	ExceptionResponseMarshaller.h	4013
src/main/activemq/wireformat/openwire/marshal/v6/	FlushCommandMarshaller.h	4017
src/main/activemq/wireformat/openwire/marshal/v6/	IntegerResponseMarshaller.h	4021
src/main/activemq/wireformat/openwire/marshal/v6/	JournalQueueAckMarshaller.h	4025
src/main/activemq/wireformat/openwire/marshal/v6/	JournalTopicAckMarshaller.h	4029
src/main/activemq/wireformat/openwire/marshal/v6/	JournalTraceMarshaller.h	4033
src/main/activemq/wireformat/openwire/marshal/v6/	JournalTransactionMarshaller.h	4037
src/main/activemq/wireformat/openwire/marshal/v6/	KeepAliveInfoMarshaller.h	4041
src/main/activemq/wireformat/openwire/marshal/v6/	LastPartialCommandMarshaller.h	4045
src/main/activemq/wireformat/openwire/marshal/v6/	LocalTransactionIdMarshaller.h	4049
src/main/activemq/wireformat/openwire/marshal/v6/	MarshallerFactory.h	4052
src/main/activemq/wireformat/openwire/marshal/v6/	MessageAckMarshaller.h	4055
src/main/activemq/wireformat/openwire/marshal/v6/	MessageDispatchMarshaller.h	4059
src/main/activemq/wireformat/openwire/marshal/v6/	MessageDispatchNotificationMarshaller.h	4064
src/main/activemq/wireformat/openwire/marshal/v6/	MessageIdMarshaller.h	4068
src/main/activemq/wireformat/openwire/marshal/v6/	MessageMarshaller.h	4072
src/main/activemq/wireformat/openwire/marshal/v6/	MessagePullMarshaller.h	4076
src/main/activemq/wireformat/openwire/marshal/v6/	NetworkBridgeFilterMarshaller.h	4080
src/main/activemq/wireformat/openwire/marshal/v6/	PartialCommandMarshaller.h	4084
src/main/activemq/wireformat/openwire/marshal/v6/	ProducerAckMarshaller.h	4088
src/main/activemq/wireformat/openwire/marshal/v6/	ProducerIdMarshaller.h	4092
src/main/activemq/wireformat/openwire/marshal/v6/	ProducerInfoMarshaller.h	4096
src/main/activemq/wireformat/openwire/marshal/v6/	RemoveInfoMarshaller.h	4100
src/main/activemq/wireformat/openwire/marshal/v6/	RemoveSubscriptionInfoMarshaller.h	4104
src/main/activemq/wireformat/openwire/marshal/v6/	ReplayCommandMarshaller.h	4108
src/main/activemq/wireformat/openwire/marshal/v6/	ResponseMarshaller.h	4112
src/main/activemq/wireformat/openwire/marshal/v6/	SessionIdMarshaller.h	4116
src/main/activemq/wireformat/openwire/marshal/v6/	SessionInfoMarshaller.h	4120
src/main/activemq/wireformat/openwire/marshal/v6/	ShutdownInfoMarshaller.h	4124

src/main/activemq/wireformat/openwire/marshall/v6/SubscriptionInfoMarshaller.h	4128
src/main/activemq/wireformat/openwire/marshall/v6/TransactionIdMarshaller.h	4132
src/main/activemq/wireformat/openwire/marshall/v6/TransactionInfoMarshaller.h	4136
src/main/activemq/wireformat/openwire/marshall/v6/WireFormatInfoMarshaller.h	4140
src/main/activemq/wireformat/openwire/marshall/v6/XATransactionIdMarshaller.h	4144
src/main/activemq/wireformat/openwire/utils/BooleanStream.h	4147
src/main/activemq/wireformat/openwire/utils/HexTable.h	4147
src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h	4148
src/main/activemq/wireformat/stomp/StompCommandConstants.h	4148
src/main/activemq/wireformat/stomp/StompFrame.h	4149
src/main/activemq/wireformat/stomp/StompHelper.h	4150
src/main/activemq/wireformat/stomp/StompWireFormat.h	4150
src/main/activemq/wireformat/stomp/StompWireFormatFactory.h	4151
src/main/cms/BytesMessage.h	4154
src/main/cms/Closeable.h	4154
src/main/cms/CMSException.h	4155
src/main/cms/CMSProperties.h	4156
src/main/cms/CMSSecurityException.h	4156
src/main/cms/Config.h	3889
src/main/cms/Connection.h	4156
src/main/cms/ConnectionFactory.h	4157
src/main/cms/ConnectionMetaData.h	4157
src/main/cms/DeliveryMode.h	4158
src/main/cms/Destination.h	4158
src/main/cms/ExceptionListener.h	4159
src/main/cms/IllegalStateException.h	4159
src/main/cms/InvalidClientIdException.h	4160
src/main/cms/InvalidDestinationException.h	4160
src/main/cms/InvalidSelectorException.h	4161
src/main/cms/MapMessage.h	4161
src/main/cms/Message.h	3830
src/main/cms/MessageConsumer.h	4162
src/main/cms/MessageEnumeration.h	4162
src/main/cms/MessageEOFException.h	4163
src/main/cms/MessageFormatException.h	4163
src/main/cms/MessageListener.h	4164
src/main/cms/MessageNotReadableException.h	4164
src/main/cms/MessageNotWriteableException.h	4164
src/main/cms/MessageProducer.h	4165
src/main/cms/ObjectMessage.h	4165
src/main/cms/Queue.h	4166
src/main/cms/QueueBrowser.h	4167
src/main/cms/Session.h	4167
src/main/cms/Startable.h	4168
src/main/cms/Stopable.h	4169
src/main/cms/StreamMessage.h	4169
src/main/cms/TemporaryQueue.h	4169
src/main/cms/TemporaryTopic.h	4170
src/main/cms/TextMessage.h	4170
src/main/cms/Topic.h	4171
src/main/cms/UnsupportedOperationException.h	4171
src/main/decaf/internal/AprPool.h	4172

src/main/decaf/internal/ DecafRuntime.h	4173
src/main/decaf/internal/io/ StandardErrorOutputStream.h	4173
src/main/decaf/internal/io/ StandardInputStream.h	4174
src/main/decaf/internal/io/ StandardOutputStream.h	4174
src/main/decaf/internal/net/ DefaultServerSocketFactory.h	4174
src/main/decaf/internal/net/ DefaultSocketFactory.h	4175
src/main/decaf/internal/net/ Network.h	4175
src/main/decaf/internal/net/ SocketFileDescriptor.h	4176
src/main/decaf/internal/net/ URIEncoderDecoder.h	4185
src/main/decaf/internal/net/ URIHelper.h	4185
src/main/decaf/internal/net/ URIType.h	4186
src/main/decaf/internal/net/ssl/ DefaultSSLContext.h	4176
src/main/decaf/internal/net/ssl/ DefaultSSLServerSocketFactory.h	4177
src/main/decaf/internal/net/ssl/ DefaultSSLSocketFactory.h	4177
src/main/decaf/internal/net/ssl/openssl/ OpenSSLContextSpi.h	4178
src/main/decaf/internal/net/ssl/openssl/ OpenSSLParameters.h	4179
src/main/decaf/internal/net/ssl/openssl/ OpenSSLServerSocket.h	4179
src/main/decaf/internal/net/ssl/openssl/ OpenSSLServerSocketFactory.h	4180
src/main/decaf/internal/net/ssl/openssl/ OpenSSLSocket.h	4180
src/main/decaf/internal/net/ssl/openssl/ OpenSSLSocketException.h	4181
src/main/decaf/internal/net/ssl/openssl/ OpenSSLSocketFactory.h	4181
src/main/decaf/internal/net/ssl/openssl/ OpenSSLSocketInputStream.h	4182
src/main/decaf/internal/net/ssl/openssl/ OpenSSLSocketOutputStream.h	4182
src/main/decaf/internal/net/tcp/ TcpSocket.h	4183
src/main/decaf/internal/net/tcp/ TcpSocketInputStream.h	4184
src/main/decaf/internal/net/tcp/ TcpSocketOutputStream.h	4184
src/main/decaf/internal/nio/ BufferFactory.h	4186
src/main/decaf/internal/nio/ ByteBuffer.h	4187
src/main/decaf/internal/nio/ CharArrayBuffer.h	4188
src/main/decaf/internal/nio/ DoubleArrayBuffer.h	4188
src/main/decaf/internal/nio/ FloatArrayBuffer.h	4189
src/main/decaf/internal/nio/ IntArrayBuffer.h	4189
src/main/decaf/internal/nio/ LongArrayBuffer.h	4190
src/main/decaf/internal/nio/ ShortArrayBuffer.h	4191
src/main/decaf/internal/security/unix/ SecureRandomImpl.h	4191
src/main/decaf/internal/security/windows/ SecureRandomImpl.h	4192
src/main/decaf/internal/util/ ByteArrayAdapter.h	4192
src/main/decaf/internal/util/ GenericResource.h	4197
src/main/decaf/internal/util/ HexStringParser.h	4198
src/main/decaf/internal/util/ Resource.h	4198
src/main/decaf/internal/util/ ResourceLifecycleManager.h	3805
src/main/decaf/internal/util/ TimerTaskHeap.h	4199
src/main/decaf/internal/util/concurrent/ ConditionImpl.h	4193
src/main/decaf/internal/util/concurrent/ MutexImpl.h	4193
src/main/decaf/internal/util/concurrent/ SynchronizableImpl.h	4194
src/main/decaf/internal/util/concurrent/ Transferer.h	4194
src/main/decaf/internal/util/concurrent/ TransferQueue.h	4195
src/main/decaf/internal/util/concurrent/ TransferStack.h	4195
src/main/decaf/internal/util/concurrent/unix/ ConditionHandle.h	4196
src/main/decaf/internal/util/concurrent/unix/ MutexHandle.h	4197
src/main/decaf/internal/util/concurrent/windows/ ConditionHandle.h	4196
src/main/decaf/internal/util/concurrent/windows/ MutexHandle.h	4197
src/main/decaf/internal/util/zip/ crc32.h	4199
src/main/decaf/internal/util/zip/ deflate.h	4199

src/main/decaf/internal/util/zip/gzguts.h	4202
src/main/decaf/internal/util/zip/inffast.h	4204
src/main/decaf/internal/util/zip/inffixed.h	4205
src/main/decaf/internal/util/zip/inflate.h	4205
src/main/decaf/internal/util/zip/inftrees.h	4206
src/main/decaf/internal/util/zip/trees.h	4207
src/main/decaf/internal/util/zip/zconf.h	4209
src/main/decaf/internal/util/zip/zlib.h	4211
src/main/decaf/internal/util/zip/zutil.h	4215
src/main/decaf/io/BlockingByteArrayInputStream.h	4217
src/main/decaf/io/BufferedInputStream.h	4218
src/main/decaf/io/BufferedOutputStream.h	4218
src/main/decaf/io/ByteArrayInputStream.h	4219
src/main/decaf/io/ByteArrayOutputStream.h	4219
src/main/decaf/io/Closeable.h	4155
src/main/decaf/io/DataInput.h	4220
src/main/decaf/io/DataInputStream.h	4220
src/main/decaf/io/DataOutput.h	4221
src/main/decaf/io/DataOutputStream.h	4222
src/main/decaf/io/EOFException.h	4222
src/main/decaf/io/FileDescriptor.h	4222
src/main/decaf/io/FilterInputStream.h	4223
src/main/decaf/io/FilterOutputStream.h	4223
src/main/decaf/io/Flushable.h	4224
src/main/decaf/io/InputStream.h	4224
src/main/decaf/io/InputStreamReader.h	4225
src/main/decaf/io/InterruptedIOException.h	4225
src/main/decaf/io/IOException.h	4226
src/main/decaf/io/OutputStream.h	4226
src/main/decaf/io/OutputStreamWriter.h	4227
src/main/decaf/io/PushbackInputStream.h	4227
src/main/decaf/io/Reader.h	4228
src/main/decaf/io/UnsupportedEncodingException.h	4228
src/main/decaf/io/UTFDataFormatException.h	4229
src/main/decaf/io/Writer.h	4229
src/main/decaf/lang/Appendable.h	4230
src/main/decaf/lang/ArrayPointer.h	4230
src/main/decaf/lang/Boolean.h	4231
src/main/decaf/lang/Byte.h	4232
src/main/decaf/lang/Character.h	4232
src/main/decaf/lang/CharSequence.h	4232
src/main/decaf/lang/Comparable.h	4233
src/main/decaf/lang/Double.h	4233
src/main/decaf/lang/Exception.h	4234
src/main/decaf/lang/Float.h	4239
src/main/decaf/lang/Integer.h	4239
src/main/decaf/lang/Iterable.h	4239
src/main/decaf/lang/Long.h	4240
src/main/decaf/lang/Math.h	4240
src/main/decaf/lang/Number.h	4241
src/main/decaf/lang/Pointer.h	4241
src/main/decaf/lang/Readable.h	4242
src/main/decaf/lang/Runnable.h	4243
src/main/decaf/lang/Runtime.h	4243

src/main/decaf/lang/	Short.h	4244
src/main/decaf/lang/	String.h	4244
src/main/decaf/lang/	System.h	4244
src/main/decaf/lang/	Thread.h	4245
src/main/decaf/lang/	ThreadGroup.h	4246
src/main/decaf/lang/	Throwable.h	4246
src/main/decaf/lang/exceptions/	ClassCastException.h	4234
src/main/decaf/lang/exceptions/	ExceptionDefines.h	3856
src/main/decaf/lang/exceptions/	IllegalArgumentException.h	4235
src/main/decaf/lang/exceptions/	IllegalMonitorStateException.h	4235
src/main/decaf/lang/exceptions/	IllegalStateException.h	4160
src/main/decaf/lang/exceptions/	IllegalThreadStateException.h	4235
src/main/decaf/lang/exceptions/	IndexOutOfBoundsException.h	4236
src/main/decaf/lang/exceptions/	InterruptedException.h	4236
src/main/decaf/lang/exceptions/	InvalidStateException.h	4237
src/main/decaf/lang/exceptions/	NoSuchElementException.h	4237
src/main/decaf/lang/exceptions/	NullPointerException.h	4237
src/main/decaf/lang/exceptions/	NumberFormatException.h	4238
src/main/decaf/lang/exceptions/	RuntimeException.h	4238
src/main/decaf/lang/exceptions/	UnsupportedOperationException.h	4172
src/main/decaf/net/	BindException.h	4247
src/main/decaf/net/	ConnectException.h	4247
src/main/decaf/net/	HttpRetryException.h	4247
src/main/decaf/net/	Inet4Address.h	4248
src/main/decaf/net/	Inet6Address.h	4248
src/main/decaf/net/	InetAddress.h	4249
src/main/decaf/net/	InetSocketAddress.h	4249
src/main/decaf/net/	MalformedURLException.h	4249
src/main/decaf/net/	NoRouteToHostException.h	4250
src/main/decaf/net/	PortUnreachableException.h	4250
src/main/decaf/net/	ProtocolException.h	4251
src/main/decaf/net/	ServerSocket.h	4251
src/main/decaf/net/	ServerSocketFactory.h	4252
src/main/decaf/net/	Socket.h	4252
src/main/decaf/net/	SocketAddress.h	4253
src/main/decaf/net/	SocketError.h	4253
src/main/decaf/net/	SocketException.h	4254
src/main/decaf/net/	SocketFactory.h	4254
src/main/decaf/net/	SocketImpl.h	4255
src/main/decaf/net/	SocketImplFactory.h	4255
src/main/decaf/net/	SocketOptions.h	4256
src/main/decaf/net/	SocketTimeoutException.h	4256
src/main/decaf/net/	UnknownHostException.h	4260
src/main/decaf/net/	UnknownServiceException.h	4260
src/main/decaf/net/	URI.h	4261
src/main/decaf/net/	URISyntaxException.h	4261
src/main/decaf/net/	URL.h	4262
src/main/decaf/net/	URLDecoder.h	4262
src/main/decaf/net/	URLEncoder.h	4262
src/main/decaf/net/ssl/	SSLContext.h	4256
src/main/decaf/net/ssl/	SSLContextSpi.h	4257
src/main/decaf/net/ssl/	SSLParameters.h	4257
src/main/decaf/net/ssl/	SSLServerSocket.h	4258
src/main/decaf/net/ssl/	SSLServerSocketFactory.h	4258

src/main/decaf/net/ssl/SSLSocket.h	4259
src/main/decaf/net/ssl/SSLSocketFactory.h	4259
src/main/decaf/nio/Buffer.h	4263
src/main/decaf/nio/BufferOverflowException.h	4263
src/main/decaf/nio/BufferUnderflowException.h	4264
src/main/decaf/nio/ByteBuffer.h	4264
src/main/decaf/nio/CharBuffer.h	4265
src/main/decaf/nio/DoubleBuffer.h	4265
src/main/decaf/nio/FloatBuffer.h	4266
src/main/decaf/nio/IntBuffer.h	4266
src/main/decaf/nio/InvalidMarkException.h	4267
src/main/decaf/nio/LongBuffer.h	4267
src/main/decaf/nio/ReadOnlyBufferException.h	4268
src/main/decaf/nio/ShortBuffer.h	4268
src/main/decaf/security/GeneralSecurityException.h	4273
src/main/decaf/security/InvalidKeyException.h	4273
src/main/decaf/security/Key.h	4273
src/main/decaf/security/KeyException.h	4274
src/main/decaf/security/KeyManagementException.h	4274
src/main/decaf/security/NoSuchAlgorithmException.h	4275
src/main/decaf/security/NoSuchProviderException.h	4275
src/main/decaf/security/Principal.h	4275
src/main/decaf/security/PublicKey.h	4276
src/main/decaf/security/SecureRandom.h	4276
src/main/decaf/security/SecureRandomSpi.h	4277
src/main/decaf/security/SignatureException.h	4277
src/main/decaf/security/auth/x500/X500Principal.h	4269
src/main/decaf/security/cert/Certificate.h	4269
src/main/decaf/security/cert/CertificateEncodingException.h	4270
src/main/decaf/security/cert/CertificateException.h	4270
src/main/decaf/security/cert/CertificateExpiredException.h	4271
src/main/decaf/security/cert/CertificateNotYetValidException.h	4271
src/main/decaf/security/cert/CertificateParsingException.h	4272
src/main/decaf/security/cert/X509Certificate.h	4272
src/main/decaf/util/AbstractCollection.h	4278
src/main/decaf/util/AbstractList.h	4278
src/main/decaf/util/AbstractMap.h	4279
src/main/decaf/util/AbstractQueue.h	4279
src/main/decaf/util/AbstractSequentialList.h	4280
src/main/decaf/util/AbstractSet.h	4281
src/main/decaf/util/Collection.h	4281
src/main/decaf/util/Comparator.h	4282
src/main/decaf/util/Config.h	3889
src/main/decaf/util/Date.h	4301
src/main/decaf/util/Iterator.h	4302
src/main/decaf/util/List.h	4302
src/main/decaf/util/ListIterator.h	4303
src/main/decaf/util/Map.h	4314
src/main/decaf/util/PriorityQueue.h	4315
src/main/decaf/util/Properties.h	4315
src/main/decaf/util/Queue.h	4166
src/main/decaf/util/Random.h	4316
src/main/decaf/util/Set.h	4316
src/main/decaf/util/StlList.h	4317

src/main/decaf/util/ StlMap.h	4318
src/main/decaf/util/ StlQueue.h	4318
src/main/decaf/util/ StlSet.h	4319
src/main/decaf/util/ StringTokenizer.h	4319
src/main/decaf/util/ Timer.h	4320
src/main/decaf/util/ TimerTask.h	4320
src/main/decaf/util/ UUID.h	4321
src/main/decaf/util/comparators/ Less.h	4282
src/main/decaf/util/concurrent/ BlockingQueue.h	4285
src/main/decaf/util/concurrent/ BrokenBarrierException.h	4285
src/main/decaf/util/concurrent/ Callable.h	4286
src/main/decaf/util/concurrent/ CancellationException.h	4286
src/main/decaf/util/concurrent/ Concurrent.h	4287
src/main/decaf/util/concurrent/ ConcurrentMap.h	4288
src/main/decaf/util/concurrent/ ConcurrentStlMap.h	4288
src/main/decaf/util/concurrent/ CountDownLatch.h	4289
src/main/decaf/util/concurrent/ Delayed.h	4289
src/main/decaf/util/concurrent/ ExecutionException.h	4290
src/main/decaf/util/concurrent/ Executor.h	4290
src/main/decaf/util/concurrent/ ExecutorService.h	4291
src/main/decaf/util/concurrent/ Future.h	4291
src/main/decaf/util/concurrent/ Lock.h	4292
src/main/decaf/util/concurrent/ Mutex.h	4295
src/main/decaf/util/concurrent/ PooledThread.h	4295
src/main/decaf/util/concurrent/ PooledThreadListener.h	4296
src/main/decaf/util/concurrent/ RejectedExecutionException.h	4296
src/main/decaf/util/concurrent/ RejectedExecutionHandler.h	4296
src/main/decaf/util/concurrent/ Semaphore.h	4297
src/main/decaf/util/concurrent/ Synchronizable.h	4297
src/main/decaf/util/concurrent/ SynchronousQueue.h	4298
src/main/decaf/util/concurrent/ TaskListener.h	4299
src/main/decaf/util/concurrent/ ThreadFactory.h	4299
src/main/decaf/util/concurrent/ ThreadPool.h	4299
src/main/decaf/util/concurrent/ TimeoutException.h	4300
src/main/decaf/util/concurrent/ TimeUnit.h	4301
src/main/decaf/util/concurrent/atomic/ AtomicBoolean.h	4283
src/main/decaf/util/concurrent/atomic/ AtomicInteger.h	4283
src/main/decaf/util/concurrent/atomic/ AtomicReferenceCounter.h	4284
src/main/decaf/util/concurrent/atomic/ AtomicReference.h	4284
src/main/decaf/util/concurrent/locks/ Condition.h	4293
src/main/decaf/util/concurrent/locks/ Lock.h	4292
src/main/decaf/util/concurrent/locks/ LockSupport.h	4293
src/main/decaf/util/concurrent/locks/ ReadWriteLock.h	4294
src/main/decaf/util/concurrent/locks/ ReentrantLock.h	4294
src/main/decaf/util/logging/ ConsoleHandler.h	4303
src/main/decaf/util/logging/ ErrorManager.h	4304
src/main/decaf/util/logging/ Filter.h	4304
src/main/decaf/util/logging/ Formatter.h	4305
src/main/decaf/util/logging/ Handler.h	4305
src/main/decaf/util/logging/ Level.h	4306
src/main/decaf/util/logging/ Logger.h	4306
src/main/decaf/util/logging/ LoggerCommon.h	4307
src/main/decaf/util/logging/ LoggerDefines.h	4307
src/main/decaf/util/logging/ LoggerHierarchy.h	4309

src/main/decaf/util/logging/ LogManager.h	4309
src/main/decaf/util/logging/ LogRecord.h	4310
src/main/decaf/util/logging/ LogWriter.h	4311
src/main/decaf/util/logging/ MarkBlockLogger.h	4311
src/main/decaf/util/logging/ PropertiesChangeListener.h	4312
src/main/decaf/util/logging/ SimpleFormatter.h	4312
src/main/decaf/util/logging/ SimpleLogger.h	4313
src/main/decaf/util/logging/ StreamHandler.h	4313
src/main/decaf/util/logging/ XMLFormatter.h	4314
src/main/decaf/util/zip/ Adler32.h	4321
src/main/decaf/util/zip/ CheckedInputStream.h	4322
src/main/decaf/util/zip/ CheckedOutputStream.h	4322
src/main/decaf/util/zip/ Checksum.h	4323
src/main/decaf/util/zip/ CRC32.h	4323
src/main/decaf/util/zip/ DataFormatException.h	4324
src/main/decaf/util/zip/ Deflater.h	4324
src/main/decaf/util/zip/ DeflaterOutputStream.h	4325
src/main/decaf/util/zip/ Inflater.h	4325
src/main/decaf/util/zip/ InflaterInputStream.h	4326
src/main/decaf/util/zip/ ZipException.h	4326

Chapter 5

Namespace Documentation

5.1 activemq Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Namespaces

- namespace **cmsutil**
- namespace **commands**
- namespace **core**
- namespace **exceptions**
- namespace **io**
- namespace **library**
- namespace **state**
- namespace **threads**
- namespace **transport**
- namespace **util**
- namespace **wireformat**

5.1.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.2 activemq::cmsutil Namespace Reference

Data Structures

- class **CachedConsumer**

A cached message consumer contained within a pooled session.

- class **CachedProducer**

A cached message producer contained within a pooled session.

- class **CmsAccessor**

*Base class for **activemq.cmsutil.CmsTemplate** (p. 1083) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1228) to operate on.*

- class **CmsDestinationAccessor**

*Extends the **CmsAccessor** (p. 1068) to add support for resolving destination names.*

- class **CmsTemplate**

***CmsTemplate** (p. 1083) simplifies performing synchronous CMS operations.*

- class **DestinationResolver**

*Resolves a CMS destination name to a **Destination**.*

- class **DynamicDestinationResolver**

*Resolves a CMS destination name to a **Destination**.*

- class **MessageCreator**

*Creates the user-defined message to be sent by the **CmsTemplate** (p. 1083).*

- class **PooledSession**

A pooled session object that wraps around a delegate session.

- class **ProducerCallback**

Callback for sending a message to a CMS destination.

- class **ResourceLifecycleManager**

Manages the lifecycle of a set of CMS resources.

- class **SessionCallback**

Callback for executing any number of operations on a provided CMS Session.

- class **SessionPool**

A pool of CMS sessions from the same connection and with the same acknowledge mode.

5.3 activemq::commands Namespace Reference

Data Structures

- class **ActiveMQBlobMessage**
- class **ActiveMQBytesMessage**
- class **ActiveMQDestination**
- class **ActiveMQMapMessage**
- class **ActiveMQMessage**
- class **ActiveMQMessageTemplate**
- class **ActiveMQObjectMessage**
- class **ActiveMQQueue**
- class **ActiveMQStreamMessage**
- class **ActiveMQTempDestination**
- class **ActiveMQTempQueue**
- class **ActiveMQTempTopic**
- class **ActiveMQTextMessage**
- class **ActiveMQTopic**
- class **BaseCommand**
- class **BaseDataStructure**
- class **BooleanExpression**
- class **BrokerError**

This class represents an Exception sent from the Broker.

- class **BrokerId**
- class **BrokerInfo**
- class **Command**
- class **ConnectionControl**
- class **ConnectionError**
- class **ConnectionId**
- class **ConnectionInfo**
- class **ConsumerControl**
- class **ConsumerId**
- class **ConsumerInfo**
- class **ControlCommand**
- class **DataArrayResponse**
- class **DataResponse**
- class **DataStructure**
- class **DestinationInfo**
- class **DiscoveryEvent**
- class **ExceptionResponse**
- class **FlushCommand**
- class **IntegerResponse**
- class **JournalQueueAck**
- class **JournalTopicAck**
- class **JournalTrace**
- class **JournalTransaction**
- class **KeepAliveInfo**
- class **LastPartialCommand**
- class **LocalTransactionId**

- class **Message**
- class **MessageAck**
- class **MessageDispatch**
- class **MessageDispatchNotification**
- class **MessageId**
- class **MessagePull**
- class **NetworkBridgeFilter**
- class **PartialCommand**
- class **ProducerAck**
- class **ProducerId**
- class **ProducerInfo**
- class **RemoveInfo**
- class **RemoveSubscriptionInfo**
- class **ReplayCommand**
- class **Response**
- class **SessionId**
- class **SessionInfo**
- class **ShutdownInfo**
- class **SubscriptionInfo**
- class **TransactionId**
- class **TransactionInfo**
- class **WireFormatInfo**
- class **XATransactionId**

5.4 activemq::core Namespace Reference

Namespaces

- namespace **policies**

Data Structures

- class **ActiveMQAckHandler**
Interface class that is used to give CMS Messages an interface to Ack themselves with.
- class **ActiveMQConnection**
Concrete connection used for all connectors to the ActiveMQ broker.
- class **ActiveMQConnectionFactory**
- class **ActiveMQConnectionMetaData**
*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 233) class.*
- class **ActiveMQConstants**
Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.
- class **ActiveMQConsumer**
- class **ActiveMQProducer**

- class **ActiveMQQueueBrowser**
- class **ActiveMQSession**
- class **ActiveMQSessionExecutor**
Delegate dispatcher for a single session.
- class **ActiveMQTransactionContext**
Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.
- class **DispatchData**
Simple POCO that contains the information necessary to route a message to a specified consumer.
- class **Dispatcher**
Interface for an object responsible for dispatching messages to consumers.
- class **MessageDispatchChannel**
- class **PrefetchPolicy**
Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.
- class **RedeliveryPolicy**
*Interface for a **RedeliveryPolicy** (p. 2972) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.*
- class **Synchronization**
*Transacted Object **Synchronization** (p. 3477), used to sync the events of a Transaction with the items in the Transaction.*

5.5 activemq::core::policies Namespace Reference

Data Structures

- class **DefaultPrefetchPolicy**
- class **DefaultRedeliveryPolicy**

5.6 activemq::exceptions Namespace Reference

Data Structures

- class **ActiveMQException**
- class **BrokerException**

5.7 activemq::io Namespace Reference

Data Structures

- class **LoggingInputStream**

- class **LoggingOutputStream**

OutputStream filter that just logs the data being written.

5.8 activemq::library Namespace Reference

Data Structures

- class **ActiveMQCPP**

5.9 activemq::state Namespace Reference

Data Structures

- class **CommandVisitor**

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

- class **CommandVisitorAdapter**

*Default Implementation of a **CommandVisitor** (p. 1113) that returns NULL for all calls.*

- class **ConnectionState**
- class **ConnectionStateTracker**
- class **ConsumerState**
- class **ProducerState**
- class **SessionState**
- class **Tracked**
- class **TransactionState**

5.10 activemq::threads Namespace Reference

Data Structures

- class **CompositeTask**

*Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p. 1133).*

- class **CompositeTaskRunner**

*A **Task** (p. 3494) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.*

- class **DedicatedTaskRunner**
- class **Task**

Represents a unit of work that requires one or more iterations to complete.

- class **TaskRunner**

5.11 activemq::transport Namespace Reference

Namespaces

- namespace **correlator**
- namespace **failover**
- namespace **inactivity**
- namespace **logging**
- namespace **mock**
- namespace **tcp**

Data Structures

- class **AbstractTransportFactory**
*Abstract implementation of the **TransportFactory** (p. 3634) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3634) instances.*
- class **CompositeTransport**
*A Composite **Transport** (p. 3629) is a **Transport** (p. 3629) implementation that is composed of several Transports.*
- class **DefaultTransportListener**
- class **IOTransport**
*Implementation of the **Transport** (p. 3629) interface that performs marshaling of commands to IO streams.*
- class **Transport**
Interface for a transport layer for command objects.
- class **TransportFactory**
Defines the interface for Factories that create Transports or TransportFilters.
- class **TransportFilter**
A filter on the transport layer.
- class **TransportListener**
A listener of asynchronous exceptions from a command transport object.
- class **TransportRegistry**
*Registry of all **Transport** (p. 3629) Factories that are available to the client at runtime.*

5.12 activemq::transport::correlator Namespace Reference

Data Structures

- class **FutureResponse**
A container that holds a response object.

- class **ResponseCorrelator**

This type of transport filter is responsible for correlating asynchronous responses with requests.

5.13 activemq::transport::failover Namespace Reference

Data Structures

- class **BackupTransport**
- class **BackupTransportPool**
- class **CloseTransportsTask**
- class **FailoverTransport**
- class **FailoverTransportFactory**

*Creates an instance of a **FailoverTransport** (p. 1752).*

- class **FailoverTransportListener**

*Utility class used by the **Transport** (p. 3629) to perform the work of responding to events from the active **Transport** (p. 3629).*

- class **URIPool**

5.14 activemq::transport::inactivity Namespace Reference

Data Structures

- class **InactivityMonitor**
- class **ReadChecker**

Runnable class that is used by the {}.

- class **WriteChecker**

Runnable class used by the {}.

5.15 activemq::transport::logging Namespace Reference

Data Structures

- class **LoggingTransport**

A transport filter that logs commands as they are sent/received.

5.16 activemq::transport::mock Namespace Reference

Data Structures

- class **InternalCommandListener**

*Listens for Commands sent from the **MockTransport** (p. 2592).*

- class **MockTransport**

*The **MockTransport** (p. 2592) defines a base level **Transport** (p. 3629) class that is intended to be used in place of an a regular protocol **Transport** (p. 3629) such as TCP.*

- class **MockTransportFactory**

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

- class **ResponseBuilder**

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

5.17 activemq::transport::tcp Namespace Reference

Data Structures

- class **SslTransport**

***Transport** (p. 3629) for connecting to a Broker using an SSL Socket.*

- class **SslTransportFactory**

- class **TcpTransport**

*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2005).*

- class **TcpTransportFactory**

*Factory Responsible for creating the **TcpTransport** (p. 3510).*

5.18 activemq::util Namespace Reference

Data Structures

- class **ActiveMQProperties**

*Implementation of the **CMSPProperties** interface that delegates to a **decaf::util::Properties** (p. 2927) object.*

- class **CMSExceptionSupport**

- class **CompositeData**

Represents a Composite URI.

- class **IdGenerator**

- class **LongSequenceGenerator**

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

- class **MarshallingSupport**

- class **MemoryUsage**
- class **PrimitiveList**
List of primitives.
- class **PrimitiveMap**
Map of named primitives.
- class **PrimitiveValueConverter**
*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2817) from one type to another.*
- class **PrimitiveValueNode**
Class that wraps around a single value of one of the many types.
- class **URISupport**
- class **Usage**

5.19 activemq::wireformat Namespace Reference

Namespaces

- namespace **openwire**
- namespace **stomp**

Data Structures

- class **MarshalAware**
- class **WireFormat**
Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.
- class **WireFormatFactory**
*The **WireFormatFactory** (p. 3716) is the interface that all **WireFormatFactory** (p. 3716) classes must extend.*
- class **WireFormatNegotiator**
*Defines a **WireFormatNegotiator** (p. 3751) which allows a **WireFormat** (p. 3712) to.*
- class **WireFormatRegistry**
*Registry of all **WireFormat** (p. 3712) Factories that are available to the client at runtime.*

5.20 activemq::wireformat::openwire Namespace Reference

Namespaces

- namespace **marshal**
- namespace **utils**

Data Structures

- class **OpenWireFormat**
- class **OpenWireFormatFactory**
- class **OpenWireFormatNegotiator**
- class **OpenWireResponseBuilder**

5.21 activemq::wireformat::openwire::marshal Namespace Reference

Namespaces

- namespace **v1**
- namespace **v2**
- namespace **v3**
- namespace **v4**
- namespace **v5**
- namespace **v6**

Data Structures

- class **BaseDataStreamMarshaller**
Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.
- class **DataStreamMarshaller**
Base class for all classes that marshal commands for Openwire.
- class **PrimitiveTypesMarshaller**
This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

5.22 activemq::wireformat::openwire::marshal::v1 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 174).*
- class **ActiveMQBytesMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 213).*
- class **ActiveMQDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 293).*

- class **ActiveMQMapMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 332).*
- class **ActiveMQMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 359).*
- class **ActiveMQObjectMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 403).*
- class **ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 445).*
- class **ActiveMQStreamMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 503).*
- class **ActiveMQTempDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 530).*
- class **ActiveMQTempQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 556).*
- class **ActiveMQTempTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 588).*
- class **ActiveMQTextMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 617).*
- class **ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 644).*
- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 714).*
- class **BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 808).*
- class **BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 839).*
- class **ConnectionControlMarshaller**
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1184).*
- class **ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1216).*
- class **ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1245).*
- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1275).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1318).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1345).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1377).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1405).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1438).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1499).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1630).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1662).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1742).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1830).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1974).*

- class **JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2036).*

- class **JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2065).*

- class **JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2087).*

- class **JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2118).*

- class **KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2145).*

- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2178).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2224).*
- class **MarshallerFactory**
Used to createmarshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2415).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2454).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2482).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2518).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2540).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2584).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2636).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2752).*
- class **ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2862).*
- class **ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2893).*
- class **ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2910).*
- class **RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3003).*
- class **RemoveSubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3019).*
- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3050).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3102).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3184).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3199).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3260).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3442).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3576).*

- class **TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3602).*

- class **WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3743).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3781).*

5.23 activemq::wireformat::openwire::marshal::v2 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 182).*

- class **ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 229).*

- class **ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 305).*

- class **ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 344).*

- class **ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 371).*

- class **ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 415).*

- class **ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 457).*

- class **ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 515).*

- class **ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 541).*

- class **ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 568).*

- class **ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 596).*

- class **ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 629).*

- class **ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 656).*

- class **BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 734).*

- class **BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 820).*

- class **BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 851).*

- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1196).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1204).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1234).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1263).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1306).*

- class **ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1334).*
- class **ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1366).*
- class **ControlCommandMarshaller**
*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1393).*
- class **DataArrayResponseMarshaller**
*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1426).*
- class **DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1487).*
- class **DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1618).*
- class **DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1651).*
- class **ExceptionResponseMarshaller**
*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1726).*
- class **FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1818).*
- class **IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1962).*
- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2021).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2049).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2072).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2102).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2129).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2166).*
- class **LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2208).*

- class **MarshallerFactory**

Used to createmarshallers for a specific version of the wire protocol.

- class **MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2403).*

- class **MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2438).*

- class **MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2470).*

- class **MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2499).*

- class **MessageMarshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2532).*

- class **MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2568).*

- class **NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2617).*

- class **PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2735).*

- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2842).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2874).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2906).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2991).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3027).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3054).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3089).*

- class **SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3165).*
- class **SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3207).*
- class **ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3256).*
- class **SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3457).*
- class **TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3580).*
- class **TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3618).*
- class **WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3735).*
- class **XATransactionIdMarshaller**
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3773).*

5.24 activemq::wireformat::openwire::marshal::v3 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 171).*
- class **ActiveMQBytesMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 210).*
- class **ActiveMQDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 289).*
- class **ActiveMQMapMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 328).*
- class **ActiveMQMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 355).*
- class **ActiveMQObjectMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 399).*

- class **ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 441).*
- class **ActiveMQStreamMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 499).*
- class **ActiveMQTempDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 527).*
- class **ActiveMQTempQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 552).*
- class **ActiveMQTempTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 580).*
- class **ActiveMQTextMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 609).*
- class **ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 636).*
- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 701).*
- class **BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 801).*
- class **BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 831).*
- class **ConnectionControlMarshaller**
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1177).*
- class **ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1208).*
- class **ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1238).*
- class **ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1267).*
- class **ConsumerControlMarshaller**
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1310).*
- class **ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1338).*

- class **ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1369).*
- class **ControlCommandMarshaller**
*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1397).*
- class **DataArrayResponseMarshaller**
*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1430).*
- class **DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1491).*
- class **DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1622).*
- class **DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1654).*
- class **ExceptionResponseMarshaller**
*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1730).*
- class **FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1822).*
- class **IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1966).*
- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2028).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2053).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2076).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2106).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2133).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2162).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2212).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.

- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2407).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2442).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2474).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2510).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2527).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2576).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2628).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2744).*
- class **ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2850).*
- class **ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2881).*
- class **ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2918).*
- class **RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2999).*
- class **RemoveSubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3023).*
- class **ReplayCommandMarshaller**
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3058).*
- class **ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3098).*
- class **SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3180).*

- class **SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3203).*
- class **ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3268).*
- class **SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3438).*
- class **TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3583).*
- class **TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3606).*
- class **WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3747).*
- class **XATransactionIdMarshaller**
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3785).*

5.25 activemq::wireformat::openwire::marshal::v4 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 178).*
- class **ActiveMQBytesMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 217).*
- class **ActiveMQDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 297).*
- class **ActiveMQMapMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 336).*
- class **ActiveMQMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 363).*
- class **ActiveMQObjectMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 407).*
- class **ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 449).*

- class **ActiveMQStreamMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 507).*
- class **ActiveMQTempDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 534).*
- class **ActiveMQTempQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 560).*
- class **ActiveMQTempTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 584).*
- class **ActiveMQTextMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 613).*
- class **ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 640).*
- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 708).*
- class **BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 805).*
- class **BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 835).*
- class **ConnectionControlMarshaller**
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1180).*
- class **ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1212).*
- class **ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1241).*
- class **ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1271).*
- class **ConsumerControlMarshaller**
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1314).*
- class **ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1342).*
- class **ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1373).*
- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1401).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1434).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1495).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1626).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1658).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1738).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1826).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1970).*

- class **JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2032).*

- class **JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2061).*

- class **JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2083).*

- class **JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2114).*

- class **KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2137).*

- class **LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2174).*

- class **LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2220).*

- class **MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

- class **MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2411).*

- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2450).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2478).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2503).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2536).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2580).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2632).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2748).*
- class **ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2846).*
- class **ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2878).*
- class **ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2902).*
- class **RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3011).*
- class **RemoveSubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3039).*
- class **ReplayCommandMarshaller**
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3046).*
- class **ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3085).*
- class **SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3169).*
- class **SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3211).*
- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3272).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3450).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3587).*

- class **TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3614).*

- class **WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3739).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3777).*

5.26 activemq::wireformat::openwire::marshal::v5 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 186).*

- class **ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 221).*

- class **ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 301).*

- class **ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 340).*

- class **ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 367).*

- class **ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 411).*

- class **ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 453).*

- class **ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 511).*

- class **ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 538).*

- class **ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 564).*

- class **ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 592).*

- class **ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 621).*

- class **ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 648).*

- class **BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 721).*

- class **BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 812).*

- class **BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 843).*

- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1188).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1220).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1249).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1279).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1322).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1349).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1381).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1409).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1442).*

- class **DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1479).*
- class **DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1638).*
- class **DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1666).*
- class **ExceptionResponseMarshaller**
*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1734).*
- class **FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1834).*
- class **IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1978).*
- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2025).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2045).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2091).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2110).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2141).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2170).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2216).*
- class **MarshallerFactory**
Used to createmarshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2419).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2446).*
- class **MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2486).*

- class **MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2507).*

- class **MessageMarshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2523).*

- class **MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2572).*

- class **NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2624).*

- class **PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2739).*

- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2854).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2885).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2914).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3007).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3035).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3066).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3093).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3176).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3195).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3264).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3446).*

- class **TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3572).*
- class **TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3598).*
- class **WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3728).*
- class **XATransactionIdMarshaller**
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3789).*

5.27 activemq::wireformat::openwire::marshal::v6 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 190).*
- class **ActiveMQBytesMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 225).*
- class **ActiveMQDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 309).*
- class **ActiveMQMapMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 348).*
- class **ActiveMQMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 375).*
- class **ActiveMQObjectMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 419).*
- class **ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 461).*
- class **ActiveMQStreamMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 519).*
- class **ActiveMQTempDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 545).*
- class **ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 572).*

- class **ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 600).*

- class **ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 625).*

- class **ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 652).*

- class **BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 728).*

- class **BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 816).*

- class **BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 847).*

- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1192).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1224).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1253).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1283).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1326).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1353).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1385).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1413).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1446).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1483).*

- class **DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1634).*
- class **DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1647).*
- class **ExceptionResponseMarshaller**
*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1722).*
- class **FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1814).*
- class **IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1958).*
- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2017).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2057).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2079).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2099).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2125).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2158).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2204).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2399).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2458).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2466).*
- class **MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2514).*

- class **MessageMarshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2544).*

- class **MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2588).*

- class **NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2620).*

- class **PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2731).*

- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2858).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2889).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2922).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2995).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3031).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3062).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3107).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3173).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3192).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3252).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3453).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3591).*

- class **TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3610).*
- class **WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3731).*
- class **XATransactionIdMarshaller**
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3769).*

5.28 activemq::wireformat::openwire::utils Namespace Reference

Data Structures

- class **BooleanStream**
*Manages the writing and reading of boolean data streams to and from a data source such as a **DataInputStream** or **DataOutputStream**.*
- class **HexTable**
*The **HexTable** (p. 1857) class maps hexadecimal strings to the value of an index into the table, i.e.*
- class **MessagePropertyInterceptor**
*Used the base **ActiveMQMessage** class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the **OpenWireMessage** properties.*

5.29 activemq::wireformat::stomp Namespace Reference

Data Structures

- class **StompCommandConstants**
- class **StompFrame**
A Stomp-level message frame that encloses all messages to and from the broker.
- class **StompHelper**
*Utility Methods used when marshaling to and from **StompFrame**'s.*
- class **StompWireFormat**
- class **StompWireFormatFactory**
Factory used to create the Stomp Wire Format instance.

5.30 cms Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Data Structures

- class **BytesMessage**

*A **BytesMessage** (p. 979) object is used to send a message containing a stream of unsigned bytes.*

- class **Closeable**

Interface for a class that implements the close method.

- class **CMSException**

CMS API Exception that is the base for all exceptions thrown from CMS classes.

- class **CMSProperties**

Interface for a Java-like properties object.

- class **CMSSecurityException**

This exception must be thrown when a provider rejects a user name/password submitted by a client.

- class **Connection**

The client's connection to its provider.

- class **ConnectionFactory**

*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1168) objects returned implement the CMS **Connection** (p. 1168) interface and hide the CMS Provider specific implementation details behind that interface.*

- class **ConnectionMetaData**

*A **ConnectionMetaData** (p. 1287) object provides information describing the **Connection** (p. 1168) object.*

- class **DeliveryMode**

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

- class **Destination**

*A **Destination** (p. 1610) object encapsulates a provider-specific address.*

- class **ExceptionListener**

*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1719) that is registered with the **Connection** (p. 1168).*

- class **IllegalStateException**

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

- class **InvalidClientIdException**

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

- class **InvalidDestinationException**

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

- class **InvalidSelectorException**

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

- class **MapMessage**

*A **MapMessage** (p. 2318) object is used to send a set of name-value pairs.*

- class **Message**

Root of all messages.

- class **MessageConsumer**

*A client uses a **MessageConsumer** (p. 2423) to received messages from a destination.*

- class **MessageEnumeration**

Defines an object that enumerates a collection of Messages.

- class **MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3415) or **BytesMessage** (p. 979) is being read.*

- class **MessageFormatException**

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

- class **MessageListener**

*A **MessageListener** (p. 2522) object is used to receive asynchronously delivered messages.*

- class **MessageNotReadableException**

This exception must be thrown when a CMS client attempts to read a write-only message.

- class **MessageNotWriteableException**

This exception must be thrown when a CMS client attempts to write to a read-only message.

- class **MessageProducer**

*A client uses a **MessageProducer** (p. 2550) object to send messages to a **Destination** (p. 1610).*

- class **ObjectMessage**

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

- class **Queue**

An interface encapsulating a provider-specific queue name.

- class **QueueBrowser**
*This class implements in interface for browsing the messages in a **Queue** (p. 2947) without removing them.*
- class **Session**
*A **Session** (p. 3148) object is a single-threaded context for producing and consuming messages.*
- class **Startable**
Interface for a class that implements the start method.
- class **Stoppable**
Interface for a class that implements the stop method.
- class **StreamMessage**
*Interface for a **StreamMessage** (p. 3415).*
- class **TemporaryQueue**
*Defines a Temporary **Queue** (p. 2947) based **Destination** (p. 1610).*
- class **TemporaryTopic**
*Defines a Temporary **Topic** (p. 3568) based **Destination** (p. 1610).*
- class **TextMessage**
Interface for a text message.
- class **Topic**
An interface encapsulating a provider-specific topic name.
- class **UnsupportedOperationException**
This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

5.30.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.31 decaf Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Namespaces

- namespace **internal**
- namespace **io**
- namespace **lang**
- namespace **net**
- namespace **nio**
- namespace **security**
- namespace **util**

5.31.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.32 decaf::internal Namespace Reference

Namespaces

- namespace **io**
- namespace **net**
- namespace **nio**
- namespace **security**
- namespace **util**

Data Structures

- class **AprPool**
Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.
- class **DecafRuntime**
Handles APR initialization and termination.

5.33 decaf::internal::io Namespace Reference

Data Structures

- class **StandardErrorOutputStream**

Wrapper Around the Standard error Output facility on the current platform.

- class **StandardInputStream**
- class **StandardOutputStream**

5.34 decaf::internal::net Namespace Reference

Namespaces

- namespace **ssl**
- namespace **tcp**

Data Structures

- class **DefaultServerSocketFactory**
Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.
- class **DefaultSocketFactory**
SocketFactory implementation that is used to create Sockets.
- class **Network**
Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.
- class **SocketFileDescriptor**
File Descriptor type used internally by Decaf Socket objects.
- class **URIEncoderDecoder**
- class **URIHelper**
Helper class used by the URI classes in encoding and decoding of URI's.
- class **URIType**
Basic type object that holds data that composes a given URI.

5.35 decaf::internal::net::ssl Namespace Reference

Namespaces

- namespace **openssl**

Data Structures

- class **DefaultSSLContext**
Default SSL Context manager for the Decaf library.
- class **DefaultSSLServerSocketFactory**

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

- class **DefaultSSLSocketFactory**

Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

5.36 decaf::internal::net::ssl::openssl Namespace Reference

Data Structures

- class **OpenSSLContextSpi**

Provides an SSLContext that wraps the OpenSSL API.

- class **OpenSSLParameters**

Container class for parameters that are Common to OpenSSL socket classes.

- class **OpenSSLServerSocket**

SSLServerSocket based on OpenSSL library code.

- class **OpenSSLServerSocketFactory**

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

- class **OpenSSLSocket**

Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

- class **OpenSSLSocketException**

Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.

- class **OpenSSLSocketFactory**

Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

- class **OpenSSLSocketInputStream**

An output stream for reading data from an OpenSSL Socket instance.

- class **OpenSSLSocketOutputStream**

*OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2673) instance.*

5.37 decaf::internal::net::tcp Namespace Reference

Data Structures

- class **TcpSocket**

Platform-independent implementation of the socket interface.

- class **TcpSocketInputStream**
Input stream for performing reads on a socket.
- class **TcpSocketOutputStream**
Output stream for performing write operations on a socket.

5.38 decaf::internal::nio Namespace Reference

Data Structures

- class **BufferFactory**
*Factory class used by static methods in the **decaf::nio** (p. 132) package to create the various default version of the NIO interfaces.*
- class **ByteArrayBuffer**
This class defines six categories of operations upon byte buffers:
 - class **CharArrayBuffer**
 - class **DoubleArrayBuffer**
 - class **FloatArrayBuffer**
 - class **IntArrayBuffer**
 - class **LongArrayBuffer**
 - class **ShortArrayBuffer**

5.39 decaf::internal::security Namespace Reference

Data Structures

- class **SecureRandomImpl**
Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

5.40 decaf::internal::util Namespace Reference

Namespaces

- namespace **concurrent**

Data Structures

- class **ByteArrayAdapter**
This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

- class **GenericResource**

*A Generic **Resource** (p. 3072) wraps some type and will delete it when the **Resource** (p. 3072) itself is deleted.*

- class **HexStringParser**
- class **Resource**

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

- class **ResourceLifecycleManager**
- class **TimerTaskHeap**

A Binary Heap implemented specifically for the Timer class in Decaf Util.

5.41 decaf::internal::util::concurrent Namespace Reference

Data Structures

- class **ConditionImpl**
- class **MutexImpl**
- class **SynchronizableImpl**

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

- class **Transferer**

Shared internal API for dual stacks and queues.

- class **TransferQueue**

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

- class **TransferStack**

5.42 decaf::io Namespace Reference

Data Structures

- class **BlockingByteArrayInputStream**

This is a blocking version of a byte buffer stream.

- class **BufferedInputStream**

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

- class **BufferedOutputStream**

Wrapper around another output stream that buffers output before writing to the target output stream.

- class **ByteArrayInputStream**

A *ByteArrayInputStream* (p. 944) contains an internal buffer that contains bytes that may be read from the stream.

- class **ByteArrayOutputStream**

- class **Closeable**

Interface for a class that implements the close method.

- class **DataInput**

The *DataInput* (p. 1452) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.

- class **DataInputStream**

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

- class **DataOutput**

The *DataOutput* (p. 1468) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.

- class **DataOutputStream**

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

- class **EOFException**

- class **FileDescriptor**

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

- class **FilterInputStream**

A *FilterInputStream* (p. 1771) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

- class **FilterOutputStream**

This class is the superclass of all classes that filter output streams.

- class **Flushable**

A *Flushable* (p. 1811) is a destination of data that can be flushed.

- class **InputStream**

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

- class **InputStreamReader**

An *InputStreamReader* (p. 1919) is a bridge from byte streams to character streams.

- class **InterruptedIOException**

- class **IOException**

- class **OutputStream**

Base interface for any class that wants to represent an output stream of bytes.

- class **OutputStreamWriter**

A class for turning a character stream into a byte stream.

- class **PushbackInputStream**

*A **PushbackInputStream** (p. 2940) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.*

- class **Reader**

- class **UnsupportedEncodingException**

Thrown when the the Character Encoding is not supported.

- class **UTFDataFormatException**

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

- class **Writer**

5.43 decaf::lang Namespace Reference

Namespaces

- namespace **exceptions**

Data Structures

- class **Appendable**

An object to which char sequences and values can be appended.

- class **ArrayPointer**

*Decaf's implementation of a Smart **Pointer** (p. 2756) that is a template on a Type and is **Thread** (p. 3520) Safe if the default Reference Counter is used.*

- class **ArrayPointerComparator**

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 669).*

- class **Boolean**

- class **Byte**

- class **Character**

- class **CharSequence**

*A **CharSequence** (p. 1053) is a readable sequence of char values.*

- class **Comparable**

This interface imposes a total ordering on the objects of each class that implements it.

- class **Double**

- class **Exception**

- class **Float**

- class **Integer**
- class **Iterable**

*Implementing this interface allows an object to be cast to an **Iterable** (p. 2011) type for generic collections API calls.*

- class **Long**
- class **Math**

*The class **Math** (p. 2339) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

- class **Number**

*The abstract class **Number** (p. 2653) is the superclass of classes **Byte** (p. 884), **Double** (p. 1672), **Float** (p. 1780), **Integer** (p. 1941), **Long** (p. 2267), and **Short** (p. 3220).*

- struct **STATIC_CAST_TOKEN**
- struct **DYNAMIC_CAST_TOKEN**
- class **Pointer**

*Decaf's implementation of a Smart **Pointer** (p. 2756) that is a template on a Type and is **Thread** (p. 3520) Safe if the default Reference Counter is used.*

- class **PointerComparator**

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2756) instance.*

- class **Readable**

*A **Readable** (p. 2958) is a source of characters.*

- class **Runnable**

Interface for a runnable object - defines a task that can be run by a thread.

- class **Runtime**
- class **Short**
- class **String**

*The **String** (p. 3427) class represents an immutable sequence of chars.*

- class **System**

*The **System** (p. 3487) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.*

- class **Thread**

*A **Thread** (p. 3520) is a concurrent unit of execution.*

- class **ThreadGroup**
- class **Throwable**

This class represents an error that has occurred.

Functions

- `template<typename T , typename R , typename U >`
`bool operator== (const ArrayPointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`
`bool operator== (const U *left, const ArrayPointer< T, R > &right)`
- `template<typename T , typename R , typename U >`
`bool operator!= (const ArrayPointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`
`bool operator!= (const U *left, const ArrayPointer< T, R > &right)`
- `template<typename T , typename R , typename U >`
`bool operator== (const Pointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`
`bool operator== (const U *left, const Pointer< T, R > &right)`
- `template<typename T , typename R , typename U >`
`bool operator!= (const Pointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`
`bool operator!= (const U *left, const Pointer< T, R > &right)`

5.43.1 Function Documentation

5.43.1.1 `template<typename T , typename R , typename U > bool`
`decaf::lang::operator!= (const ArrayPointer< T, R > & left, const U * right) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

5.43.1.2 `template<typename T , typename R , typename U > bool`
`decaf::lang::operator!= (const U * left, const ArrayPointer< T, R > & right) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

5.43.1.3 `template<typename T , typename R , typename U > bool`
`decaf::lang::operator!= (const U * left, const Pointer< T, R > & right) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.43.1.4 `template<typename T , typename R , typename U > bool`
`decaf::lang::operator!= (const Pointer< T, R > & left, const U * right) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.43.1.5 `template<typename T , typename R , typename U > bool`
`decaf::lang::operator== (const Pointer< T, R > & left, const U * right) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.43.1.6 `template<typename T , typename R , typename U > bool
decaf::lang::operator==(const U * left, const ArrayPointer< T, R > &
right) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

5.43.1.7 `template<typename T , typename R , typename U > bool
decaf::lang::operator==(const U * left, const Pointer< T, R > &
right) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.43.1.8 `template<typename T , typename R , typename U > bool
decaf::lang::operator==(const ArrayPointer< T, R > & left, const U *
right) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

5.44 decaf::lang::exceptions Namespace Reference

Data Structures

- class `ClassCastException`
- class `IllegalArgumentException`
- class `IllegalMonitorStateException`
- class `IllegalStateException`
- class `IllegalThreadStateException`
- class `IndexOutOfBoundsException`
- class `InterruptedException`
- class `InvalidStateException`
- class `NoSuchElementException`
- class `NullPointerException`
- class `NumberFormatException`
- class `RuntimeException`
- class `UnsupportedOperationException`

5.45 decaf::net Namespace Reference

Namespaces

- namespace `ssl`

Data Structures

- class `BindException`
- class `ConnectException`
- class `HttpRetryException`

- class **Inet4Address**
- class **Inet6Address**
- class **InetAddress**

Represents an IP address.

- class **InetSocketAddress**
- class **MalformedURLException**
- class **NoRouteToHostException**
- class **PortUnreachableException**
- class **ProtocolException**
- class **ServerSocket**

This class implements server sockets.

- class **ServerSocketFactory**

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

- class **Socket**
- class **SocketAddress**

*Base class for protocol specific **Socket** (p. 3281) addresses.*

- class **SocketError**

Static utility class to simplify handling of error codes for socket operations.

- class **SocketException**

Exception for errors when manipulating sockets.

- class **SocketFactory**

*The **SocketFactory** (p. 3301) is used to create **Socket** (p. 3281) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*

- class **SocketImpl**

*Acts as a base class for all physical **Socket** (p. 3281) implementations.*

- class **SocketImplFactory**

*Factory class interface for a Factory that creates **SocketImpl** objects.*

- class **SocketOptions**
- class **SocketTimeoutException**
- class **UnknownHostException**
- class **UnknownServiceException**
- class **URI**

*This class represents an instance of a **URI** (p. 3660) as defined by RFC 2396.*

- class **URISyntaxException**
- class **URL**

*Class **URL** (p. 3697) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

- class **URLDecoder**
- class **URLEncoder**

5.46 decaf::net::ssl Namespace Reference

Data Structures

- class **SSLContext**
*Represents an implementation of the Secure **Socket** (p. 3281) Layer for streaming based sockets.*
- class **SSLContextSpi**
*Defines the interface that should be provided by an **SSLContext** (p. 3321) provider.*
- class **SSLParameters**
- class **SSLServerSocket**
Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.
- class **SSLServerSocketFactory**
Factory class interface that provides methods to create SSL Server Sockets.
- class **SSLSocket**
- class **SSLSocketFactory**
*Factory class interface for a **SocketFactory** (p. 3301) that can create **SSLSocket** (p. 3337) objects.*

5.47 decaf::nio Namespace Reference

Data Structures

- class **Buffer**
A container for data of a specific primitive type.
- class **BufferOverflowException**
- class **BufferUnderflowException**
- class **ByteBuffer**
This class defines six categories of operations upon byte buffers:
- class **CharBuffer**
This class defines four categories of operations upon character buffers:
- class **DoubleBuffer**
This class defines four categories of operations upon double buffers:
- class **FloatBuffer**
This class defines four categories of operations upon float buffers:
- class **IntBuffer**
This class defines four categories of operations upon int buffers:
- class **InvalidMarkException**

- class **LongBuffer**

This class defines four categories of operations upon long long buffers:

- class **ReadOnlyBufferException**
- class **ShortBuffer**

This class defines four categories of operations upon short buffers:

5.48 decaf::security Namespace Reference

Namespaces

- namespace **auth**
- namespace **cert**

Data Structures

- class **GeneralSecurityException**
- class **InvalidKeyException**
- class **Key**

*The **Key** (p. 2149) interface is the top-level interface for all keys.*

- class **KeyException**
- class **KeyManagementException**
- class **NoSuchAlgorithmException**
- class **NoSuchProviderException**
- class **Principal**

Base interface for a principal, which can represent an individual or organization.

- class **PublicKey**

A public key.

- class **SecureRandom**
- class **SecureRandomSpi**

Interface class used by Security Service Providers to implement a source of secure random bytes.

- class **SignatureException**

5.49 decaf::security::auth Namespace Reference

Namespaces

- namespace **x500**

5.50 decaf::security::auth::x500 Namespace Reference

Data Structures

- class **X500Principal**

5.51 decaf::security::cert Namespace Reference

Data Structures

- class **Certificate**
Base interface for all identity certificates.
- class **CertificateEncodingException**
- class **CertificateException**
- class **CertificateExpiredException**
- class **CertificateNotYetValidException**
- class **CertificateParsingException**
- class **X509Certificate**
Base interface for all identity certificates.

5.52 decaf::util Namespace Reference

Namespaces

- namespace **comparators**
- namespace **concurrent**
- namespace **logging**
- namespace **zip**

Data Structures

- class **AbstractCollection**
*This class provides a skeletal implementation of the **Collection** (p. 1097) interface, to minimize the effort required to implement this interface.*
- class **AbstractList**
*This class provides a skeletal implementation of the **List** (p. 2190) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*
- class **AbstractMap**
*This class provides a skeletal implementation of the **Map** (p. 2305) interface, to minimize the effort required to implement this interface.*
- class **AbstractQueue**
*This class provides skeletal implementations of some **Queue** (p. 2948) operations.*

- class **AbstractSequentialList**

*This class provides a skeletal implementation of the **List** (p. 2190) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

- class **AbstractSet**

*This class provides a skeletal implementation of the **Set** (p. 3220) interface to minimize the effort required to implement this interface.*

- class **Collection**

The root interface in the collection hierarchy.

- class **Comparator**

A comparison function, which imposes a total ordering on some collection of objects.

- class **Date**

Wrapper class around a time value in milliseconds.

- class **Iterator**

Defines an object that can be used to iterate over the elements of a collection.

- class **List**

An ordered collection (also known as a sequence).

- class **ListIterator**

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

- class **Map**

***Map** (p. 2305) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*

- class **PriorityQueue**

An unbounded priority queue based on a binary heap algorithm.

- class **Properties**

Java-like properties class for mapping string names to string values.

- class **Queue**

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

- class **Random**

***Random** (p. 2953) Value Generator which is used to generate a stream of pseudorandom numbers.*

- class **Set**

A collection that contains no duplicate elements.

- class **StIList**

***List** (p. 2190) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.*

- class **StlMap**

***Map** (p. 2305) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*

- class **StlQueue**

*The **Queue** (p. 2948) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.*

- class **StlSet**

***Set** (p. 3220) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.*

- class **StringTokenizer**

- class **Timer**

A facility for threads to schedule tasks for future execution in a background thread.

- class **TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3543).*

- class **UUID**

*A class that represents an immutable universally unique identifier (**UUID** (p. 3706)).*

5.53 decaf::util::comparators Namespace Reference

Data Structures

- class **Less**

*Simple **Less** (p. 2182) **Comparator** (p. 1127) that compares to elements to determine if the first is less than the second.*

5.54 decaf::util::concurrent Namespace Reference

Namespaces

- namespace **atomic**
- namespace **locks**

Data Structures

- class **ConditionHandle**
- class **MutexHandle**
- class **BlockingQueue**

A *decaf::util::Queue* (p. 2948) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

- class **BrokenBarrierException**
- class **Callable**

A task that returns a result and may throw an exception.

- class **CancellationException**
- class **ConcurrentMap**

Interface for a **Map** (p. 2305) type that provides additional atomic *putIfAbsent*, *remove*, and *replace* methods alongside the already available **Map** (p. 2305) interface.

- class **ConcurrentStlMap**

Map (p. 2305) template that wraps around a *std::map* to provide a more user-friendly interface and to provide common functions that do not exist in *std::map*.

- class **CountDownLatch**
- class **Delayed**

A mix-in style interface for marking objects that should be acted upon after a given delay.

- class **ExecutionException**
- class **Executor**

An object that executes submitted *decaf.lang Runnable* (p. 3111) tasks.

- class **ExecutorService**

An **Executor** (p. 1749) that provides methods to manage termination and methods that can produce a **Future** (p. 1840) for tracking progress of one or more asynchronous tasks.

- class **Future**

A **Future** (p. 1840) represents the result of an asynchronous computation.

- class **Lock**

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

- class **Mutex**

Mutex (p. 2604) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

- class **PooledThread**
- class **PooledThreadListener**

Abstract Listener Interface for users of *ThreadPool* (p. 3531).

- class **RejectedExecutionException**
- class **RejectedExecutionHandler**

A handler for tasks that cannot be executed by a *ThreadPoolExecutor* (p. ??).

- class **Semaphore**

A counting semaphore.

- class **Synchronizable**

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

- class **SynchronousQueue**

*A **blocking queue** (p. 774) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.*

- class **TaskListener**

- class **ThreadFactory**

*public interface **ThreadFactory** (p. 3529)*

- class **ThreadPool**

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

- class **TimeoutException**

- class **TimeUnit**

*A **TimeUnit** (p. 3559) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

5.55 decaf::util::concurrent::atomic Namespace Reference

Data Structures

- class **AtomicBoolean**

A boolean value that may be updated atomically.

- class **AtomicInteger**

An int value that may be updated atomically.

- class **AtomicRefCounter**

- class **AtomicReference**

An Pointer reference that may be updated atomically.

5.56 decaf::util::concurrent::locks Namespace Reference

Data Structures

- class **Condition**

***Condition** (p. 1156) factors out the **Mutex** (p. 2604) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2229) implementations.*

- class **Lock**

***Lock** (p. 2229) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

- class **LockSupport**
Basic thread blocking primitives for creating locks and other synchronization classes.
- class **ReadWriteLock**
*A **ReadWriteLock** (p. 2968) maintains a pair of associated locks, one for read-only operations and one for writing.*
- class **ReentrantLock**
*A reentrant mutual exclusion **Lock** (p. 2229) with extended capabilities.*

5.57 decaf::util::logging Namespace Reference

Data Structures

- class **ConsoleHandler**
*This **Handler** (p. 1852) publishes log records to `System.err`.*
- class **ErrorManager**
***ErrorManager** (p. 1710) objects can be attached to **Handlers** to process any error that occur on a **Handler** (p. 1852) during Logging.*
- class **Filter**
*A **Filter** (p. 1770) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*
- class **Formatter**
*A **Formatter** (p. 1838) provides support for formatting `LogRecords`.*
- class **Handler**
*A **Handler** (p. 1852) object takes log messages from a **Logger** (p. 2237) and exports them.*
- class **Level**
*The **Level** (p. 2185) class defines a set of standard logging levels that can be used to control logging output.*
- class **Logger**
*A **Logger** (p. 2237) object is used to log messages for a specific system or application component.*
- class **LoggerHierarchy**
- class **LogManager**
*There is a single global **LogManager** (p. 2254) object that is used to maintain a set of shared state about `Loggers` and log services.*
- class **LogRecord**
***LogRecord** (p. 2261) objects are used to pass logging requests between the logging framework and individual log `Handlers`.*
- class **LogWriter**

- class **MarkBlockLogger**

Defines a class that can be used to mark the entry and exit from scoped blocks.

- class **PropertiesChangeListener**

*Defines the interface that classes can use to listen for change events on **Properties** (p. 2927).*

- class **SimpleFormatter**

*Print a brief summary of the **LogRecord** (p. 2261) in a human readable format.*

- class **SimpleLogger**

- class **StreamHandler**

*Stream based logging **Handler** (p. 1852).*

- class **XMLFormatter**

*Format a **LogRecord** (p. 2261) into a standard XML format.*

Enumerations

- enum **Levels** {

Off, **Null**, **Markblock**, **Debug**,

Info, **Warn**, **Error**, **Fatal**,

Throwing }

Defines an enumeration for logging levels.

5.57.1 Enumeration Type Documentation

5.57.1.1 enum decaf::util::logging::Levels

Defines an enumeration for logging levels.

Enumerator:

Off

Null

Markblock

Debug

Info

Warn

Error

Fatal

Throwing

5.58 decaf::util::zip Namespace Reference

Data Structures

- class **Adler32**
*Class that can be used to compute an Adler-32 **Checksum** (p. 1059) for a data stream.*
- class **CheckedInputStream**
*An implementation of a **FilterInputStream** that will maintain a **Checksum** (p. 1059) of the bytes read, the **Checksum** (p. 1059) can then be used to verify the integrity of the input stream.*
- class **CheckedOutputStream**
*An implementation of a **FilterOutputStream** that will maintain a **Checksum** (p. 1059) of the bytes written, the **Checksum** (p. 1059) can then be used to verify the integrity of the output stream.*
- class **Checksum**
*An interface used to represent **Checksum** (p. 1059) values in the Zip package.*
- class **CRC32**
Class that can be used to compute a CRC-32 checksum for a data stream.
- class **DataFormatException**
- class **Deflater**
*This class compresses data using the DEFLATE algorithm (see **specification**).*
- class **DeflaterOutputStream**
*Provides a **FilterOutputStream** instance that compresses the data before writing it to the wrapped **OutputStream**.*
- class **Inflater**
*This class uncompresses data that was compressed using the DEFLATE algorithm (see **specification**).*
- class **InflaterInputStream**
*A **FilterInputStream** that decompresses data read from the wrapped **InputStream** instance.*
- class **ZipException**

5.59 std Namespace Reference

Data Structures

- struct **less< decaf::lang::ArrayPointer< T > >**
*An override of the less function object so that the **Pointer** objects can be stored in STL Maps, etc.*
- struct **less< decaf::lang::Pointer< T > >**
*An override of the less function object so that the **Pointer** objects can be stored in STL Maps, etc.*

Chapter 6

Data Structure Documentation

6.1 decaf::util::AbstractCollection< E > Class Template Reference

This class provides a skeletal implementation of the **Collection** (p.1097) interface, to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractCollection.h>
```

Inheritance diagram for decaf::util::AbstractCollection< E >:

Public Member Functions

- **AbstractCollection** ()
- virtual ~**AbstractCollection** ()
- **AbstractCollection**< E > & **operator=** (const **AbstractCollection**< E > &collection)

Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.

- virtual bool **add** (const E &value DECAF_UNUSED)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

Ensures that this collection contains the specified element (optional operation).

- virtual bool **addAll** (const **Collection**< E > &collection)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

Adds all of the elements in the specified collection to this collection (optional operation).

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)

Removes all of the elements from this collection (optional operation).

- virtual void **copy** (const **Collection**< E > &collection)

*Renders this **Collection** (p. 1097) as a Copy of the given **Collection** (p. 1097).*

- virtual bool **contains** (const E &value) const throw (lang::Exception)
Returns true if this collection contains the specified element.
- virtual bool **containsAll** (const **Collection**< E > &collection) const throw (lang::Exception)
Returns true if this collection contains all of the elements in the specified collection.
- virtual bool **equals** (const **Collection**< E > &collection) const
*Answers true if this **Collection** (p. 1097) and the one given are the same size and if each element contained in the **Collection** (p. 1097) given is equal to an element contained in this collection.*
- virtual bool **isEmpty** () const
Returns true if this collection contains no elements.
- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes a single instance of the specified element from this collection, if it is present (optional operation).
- virtual bool **removeAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes all of this collection's elements that are also contained in the specified collection (optional operation).
- virtual bool **retainAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Retains only the elements in this collection that are contained in the specified collection (optional operation).
- virtual std::vector< E > **toArray** () const
*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1097).*
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

Protected Attributes

- util::concurrent::Mutex mutex

6.1.1 Detailed Description

template<typename E> class decaf::util::AbstractCollection< E >

This class provides a skeletal implementation of the **Collection** (p. 1097) interface, to minimize the effort required to implement this interface. To implement an unmodifiable collection, the programmer needs only to extend this class and provide implementations for the iterator and size methods. (The iterator returned by the iterator method must implement hasNext and next.)

To implement a modifiable collection, the programmer must additionally override this class's add method (which otherwise throws an UnsupportedOperationException), and the iterator returned by the iterator method must additionally implement its remove method.

The programmer should generally provide a void (no argument) and **Collection** (p. 1097) constructor, as per the recommendation in the **Collection** (p. 1097) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

Since

1.0

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `template<typename E> decaf::util::AbstractCollection< E
>::AbstractCollection () [inline]`

6.1.2.2 `template<typename E> virtual decaf::util::AbstractCollection< E
>::~AbstractCollection () [inline, virtual]`

6.1.3 Member Function Documentation

6.1.3.1 `template<typename E> virtual bool decaf::util::AbstractCollection<
E >::add (const E &value DECAF_UNUSED)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException,
lang::exceptions::IllegalStateException) [inline,
virtual]`

Ensures that this collection contains the specified element (optional operation).

Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. **Collection** (p. 1097) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

This implementation always throws an `UnsupportedOperationException`.

Parameters

value - The element that must be ensured to be in this collection.

Returns

true if the collection was changed as a result of this call.

Exceptions

UnsupportedOperationException if the add operation is not supported by this collection

IllegalArgumentException if some property of the element prevents it from being added to this collection

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::addAll()`, `decaf::util::AbstractCollection< cms::Connection * >::copy()`, and `decaf::util::AbstractCollection< cms::Connection * >::operator=()`.

```

6.1.3.2  template<typename E> virtual bool decaf::util::AbstractCollection<
        E >::addAll ( const Collection< E > & collection )
        throw ( lang::exceptions::UnsupportedOperationException,
        lang::exceptions::IllegalArgumentException,
        lang::exceptions::IllegalStateException ) [inline,
        virtual]

```

Adds all of the elements in the specified collection to this collection (optional operation).

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

Parameters

collection - The **Collection** (p.1097) whose elements are to be added to this **Collection** (p.1097).

Returns

true if the collection was changed as a result of this call.

Exceptions

UnsupportedOperationException if the `addAll` operation is not supported by this collection

IllegalArgumentException if some property of the element prevents it from being added to this collection

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p.1100).

Reimplemented in **decaf::util::AbstractQueue< E >** (p.160).

```

6.1.3.3  template<typename E> virtual void decaf::util::AbstractCollection< E
        >::clear ( ) throw ( lang::exceptions::UnsupportedOperationException )
        [inline, virtual]

```

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p.2014) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the `remove` method and this collection is non-empty.

Exceptions

UnsupportedOperationException if the clear operation is not supported by this collection

Implements **decaf::util::Collection**< **E** > (p. 1100).

Reimplemented in **decaf::util::AbstractQueue**< **E** > (p. 160), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 3480), **decaf::util::PriorityQueue**< **E** > (p. 2836), **decaf::util::StlList**< **E** > (p. 3364), **decaf::util::StlSet**< **E** > (p. 3393), **decaf::util::StlList**< **cms::MessageConsumer** * > (p. 3364), **decaf::util::StlList**< **CompositeTask** * > (p. 3364), **decaf::util::StlList**< **URI** > (p. 3364), **decaf::util::StlList**< **Pointer**< **DestinationInfo** > > (p. 3364), **decaf::util::StlList**< **PrimitiveValueNode** > (p. 3364), **decaf::util::StlList**< **Pointer**< **Command** > > (p. 3364), **decaf::util::StlList**< **Pointer**< **BackupTransport** > > (p. 3364), **decaf::util::StlList**< **cms::MessageProducer** * > (p. 3364), **decaf::util::StlList**< **cms::Destination** * > (p. 3364), **decaf::util::StlList**< **cms::Session** * > (p. 3364), **decaf::util::StlList**< **cms::Connection** * > (p. 3364), **decaf::util::StlSet**< **transport::TransportListener** * > (p. 3393), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 3393), **decaf::util::StlSet**< **Resource** * > (p. 3393), and **decaf::util::StlSet**< **ActiveMQSession** * > (p. 3393).

Referenced by **decaf::util::AbstractCollection**< **cms::Connection** * >::copy(), and **decaf::util::AbstractCollection**< **cms::Connection** * >::operator=().

6.1.3.4 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::contains (const E & value) const throw (lang::Exception) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters

value - the value whose presence is to be queried for in this **Collection** (p. 1097).

Returns

true if the value is contained in this collection

Exceptions

Exception if an error occurs,

Implements **decaf::util::Collection**< **E** > (p. 1101).

Reimplemented in **decaf::util::StlList**< **E** > (p. 3365), **decaf::util::StlSet**< **E** > (p. 3394), **decaf::util::StlList**< **cms::MessageConsumer** * > (p. 3365), **decaf::util::StlList**< **CompositeTask** * > (p. 3365), **decaf::util::StlList**< **URI** > (p. 3365), **decaf::util::StlList**< **Pointer**< **DestinationInfo** > > (p. 3365), **decaf::util::StlList**< **PrimitiveValueNode** > (p. 3365), **decaf::util::StlList**< **Pointer**< **Command** > > (p. 3365), **decaf::util::StlList**< **Pointer**< **BackupTransport** > > (p. 3365), **decaf::util::StlList**< **cms::MessageProducer** * > (p. 3365), **decaf::util::StlList**< **cms::Destination** * > (p. 3365), **decaf::util::StlList**< **cms::Session** * > (p. 3365), **decaf::util::StlList**< **cms::Connection** * > (p. 3365), **decaf::util::StlSet**< **transport::TransportListener** * > (p. 3394), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 3394), **decaf::util::StlSet**< **Resource** * > (p. 3394), and **decaf::util::StlSet**< **ActiveMQSession** * > (p. 3394).

Referenced by **decaf::util::AbstractCollection**< **cms::Connection** * >::containsAll().

6.1.3.5 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::containsAll (const Collection< E > & collection) const throw (lang::Exception) [inline, virtual]`

Returns true if this collection contains all of the elements in the specified collection.

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

Parameters

collection collection to be checked for containment in this collection

Returns

true if this collection contains all of the elements in the specified collection.

Exceptions

Exception if an error occurs,

Implements `decaf::util::Collection< E >` (p. 1102).

Reimplemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3480).

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::equals()`.

6.1.3.6 `template<typename E> virtual void decaf::util::AbstractCollection< E >::copy (const Collection< E > & collection) [inline, virtual]`

Renders this **Collection** (p. 1097) as a Copy of the given **Collection** (p. 1097).

This implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection**'s clear method.

Parameters

collection - the collection to mirror.

6.1.3.7 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::equals (const Collection< E > & collection) const [inline, virtual]`

Answers true if this **Collection** (p. 1097) and the one given are the same size and if each element contained in the **Collection** (p. 1097) given is equal to an element contained in this collection.

Parameters

collection - The **Collection** (p. 1097) to be compared to this one.

Returns

true if this **Collection** (p. 1097) is equal to the one given.

Implements `decaf::util::Collection< E >` (p. 1103).

Reimplemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3482).

6.1.3.8 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::isEmpty () const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns `size() (p. 1106) == 0`.

Returns

true if the size method return 0.

Implements `decaf::util::Collection< E > (p. 1103)`.

Reimplemented in `decaf::util::concurrent::SynchronousQueue< E > (p. 3482)`, `decaf::util::StlList< E > (p. 3366)`, `decaf::util::StlSet< E > (p. 3394)`, `decaf::util::StlList< cms::MessageConsumer * > (p. 3366)`, `decaf::util::StlList< CompositeTask * > (p. 3366)`, `decaf::util::StlList< URI > (p. 3366)`, `decaf::util::StlList< Pointer< DestinationInfo > > (p. 3366)`, `decaf::util::StlList< PrimitiveValueNode > (p. 3366)`, `decaf::util::StlList< Pointer< Command > > (p. 3366)`, `decaf::util::StlList< Pointer< BackupTransport > > (p. 3366)`, `decaf::util::StlList< cms::MessageProducer * > (p. 3366)`, `decaf::util::StlList< cms::Destination * > (p. 3366)`, `decaf::util::StlList< cms::Session * > (p. 3366)`, `decaf::util::StlList< cms::Connection * > (p. 3366)`, `decaf::util::StlSet< transport::TransportListener * > (p. 3394)`, `decaf::util::StlSet< Pointer< Synchronization > > (p. 3394)`, `decaf::util::StlSet< Resource * > (p. 3394)`, and `decaf::util::StlSet< ActiveMQSession * > (p. 3394)`.

Referenced by `decaf::util::AbstractQueue< E >::clear()`.

6.1.3.9 `template<typename E> virtual void decaf::util::AbstractCollection< E >::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable (p. 3463)`.

6.1.3.10 `template<typename E> virtual void decaf::util::AbstractCollection< E >::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable (p. 3464)`.

6.1.3.11 `template<typename E> virtual void decaf::util::AbstractCollection< E >::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3465).

6.1.3.12 `template<typename E> AbstractCollection<E>& decaf::util::AbstractCollection< E >::operator= (const AbstractCollection< E > & collection) [inline]`

Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.

Parameters

collection - the collection to copy

Returns

a reference to this collection

6.1.3.13 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 1097).

Implements **decaf::util::Collection< E >** (p. 1104).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 2838), **decaf::util::StlList< E >** (p. 3368), **decaf::util::StlSet< E >** (p. 3394), **decaf::util::StlList< cms::MessageConsumer * >** (p. 3368), **decaf::util::StlList< CompositeTask * >** (p. 3368), **decaf::util::StlList< URI >** (p. 3368), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3368), **decaf::util::StlList< PrimitiveValueNode >** (p. 3368), **decaf::util::StlList< Pointer< Command > >** (p. 3368), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3368), **decaf::util::StlList< cms::MessageProducer * >** (p. 3368), **decaf::util::StlList< cms::Destination * >** (p. 3368), **decaf::util::StlList< cms::Session * >** (p. 3368), **decaf::util::StlList< cms::Connection * >** (p. 3368), **decaf::util::StlSet< transport::TransportListener * >** (p. 3394), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3394), **decaf::util::StlSet< Resource * >** (p. 3394), and **decaf::util::StlSet< ActiveMQSession * >** (p. 3394).

```
6.1.3.14  template<typename E> virtual bool decaf::util::AbstractCollection<
           E >::removeAll ( const Collection< E > & collection )
           throw ( lang::exceptions::UnsupportedOperationException,
                   lang::exceptions::IllegalArgumentException ) [inline, virtual]
```

Removes all of this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

Parameters

collection - collection containing elements to be removed from this collection

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection

IllegalArgumentException.

Implements **decaf::util::Collection< E >** (p. 1105).

Reimplemented in `decaf::util::AbstractSet< E >` (p.163), `decaf::util::AbstractSet< transport::TransportListener * >` (p.163), `decaf::util::AbstractSet< Pointer< Synchronization > >` (p.163), `decaf::util::AbstractSet< Resource * >` (p.163), and `decaf::util::AbstractSet< ActiveMQSession * >` (p.163).

```
6.1.3.15  template<typename E> virtual bool decaf::util::AbstractCollection<
           E >::retainAll (  const Collection< E > &  collection  )
           throw ( lang::exceptions::UnsupportedOperationException,
                   lang::exceptions::IllegalArgumentException ) [inline, virtual]
```

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

Parameters

collection - collection containing elements to be retained in this collection

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection

IllegalArgumentException.

Implements `decaf::util::Collection< E >` (p.1105).

```
6.1.3.16  template<typename E> virtual std::vector<E>
           decaf::util::AbstractCollection< E >::toArray (    ) const [inline,
           virtual]
```

Answers an STL vector containing copies of all elements contained in this **Collection** (p.1097).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

Returns

an vector of copies of all the elements from this **Collection** (p.1097)

Implements `decaf::util::Collection< E >` (p.1107).

Reimplemented in `decaf::util::concurrent::SynchronousQueue< E >` (p.3486).

6.1.3.17 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::tryLock () throw (decaf::lang::exceptions::RuntimeException)`
`[inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3466).

6.1.3.18 `template<typename E> virtual void decaf::util::AbstractCollection< E >::unlock () throw (decaf::lang::exceptions::RuntimeException)`
`[inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3467).

6.1.3.19 `template<typename E> virtual void decaf::util::AbstractCollection< E >::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)` `[inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3468).

6.1.3.20 `template<typename E> virtual void decaf::util::AbstractCollection< E >::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentOutOfRangeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)` `[inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3471).

```
6.1.3.21  template<typename E> virtual void decaf::util::AbstractCollection<
           E >::wait ( long long  milliseconds ) throw (
           decaf::lang::exceptions::RuntimeException, de-
           caf::lang::exceptions::IllegalMonitorStateException,
           decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3470).

6.1.4 Field Documentation

```
6.1.4.1  template<typename E> util::concurrent::Mutex
           decaf::util::AbstractCollection< E >::mutex [mutable, protected]
```

Referenced by decaf::util::AbstractCollection< cms::Connection * >::lock(),
decaf::util::AbstractCollection< cms::Connection * >::notify(), decaf::util::AbstractCollection<

```
cms::Connection * >::notifyAll(), decaf::util::AbstractCollection< cms::Connection
* >::tryLock(), decaf::util::AbstractCollection< cms::Connection * >::unlock(), and
decaf::util::AbstractCollection< cms::Connection * >::wait().
```

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractCollection.h`

6.2 decaf::util::AbstractList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 2190) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).

```
#include <src/main/decaf/util/AbstractList.h>
```

Inheritance diagram for `decaf::util::AbstractList< E >`:

Public Member Functions

- `virtual ~AbstractList ()`

6.2.1 Detailed Description

```
template<typename E> class decaf::util::AbstractList< E >
```

This class provides a skeletal implementation of the **List** (p. 2190) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array). For sequential access data (such as a linked list), **AbstractSequentialList** (p. 161) should be used in preference to this class.

To implement an unmodifiable list, the programmer needs only to extend this class and provide implementations for the `get(int)` and `size()` (p. 1106) methods.

To implement a modifiable list, the programmer must additionally override the `set(int, E)` method (which otherwise throws an `UnsupportedOperationException`). If the list is variable-size the programmer must additionally override the `add(int, E)` and `remove(int)` methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 1097) interface specification.

Unlike the other abstract collection implementations, the programmer does not have to provide an iterator implementation; the iterator and list iterator are implemented by this class, on top of the "random access" methods: `get(int)`, `set(int, E)`, `add(int, E)` and `remove(int)`.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

Since

1.0

6.2.2 Constructor & Destructor Documentation

6.2.2.1 `template<typename E > virtual decaf::util::AbstractList< E
>::~AbstractList () [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractList.h`

6.3 decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference

This class provides a skeletal implementation of the **Map** (p. 2305) interface, to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractMap.h>
```

Inheritance diagram for `decaf::util::AbstractMap< K, V, COMPARATOR >`:

Public Member Functions

- `virtual ~AbstractMap ()`

6.3.1 Detailed Description

`template<typename K, typename V, typename COMPARATOR> class
decaf::util::AbstractMap< K, V, COMPARATOR >`

This class provides a skeletal implementation of the **Map** (p. 2305) interface, to minimize the effort required to implement this interface. To implement an unmodifiable map, the programmer needs only to extend this class and provide an implementation for the `entrySet` method, which returns a set-view of the map's mappings. Typically, the returned set will, in turn, be implemented atop **AbstractSet** (p. 162). This set should not support the `add` or `remove` methods, and its iterator should not support the `remove` method.

To implement a modifiable map, the programmer must additionally override this class's `put` method (which otherwise throws an `UnsupportedOperationException`), and the iterator returned by `entrySet().iterator()` must additionally implement its `remove` method.

The programmer should generally provide a void (no argument) and map constructor, as per the recommendation in the **Map** (p. 2305) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the map being implemented admits a more efficient implementation.

Since

1.0

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `template<typename K , typename V , typename COMPARATOR >
virtual decaf::util::AbstractMap< K, V, COMPARATOR >::~~AbstractMap
() [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractMap.h`

6.4 decaf::util::AbstractQueue< E > Class Template Reference

This class provides skeletal implementations of some **Queue** (p. 2948) operations.

`#include <src/main/decaf/util/AbstractQueue.h>`

Inheritance diagram for `decaf::util::AbstractQueue< E >`:

Public Member Functions

- **AbstractQueue** ()
- virtual **~AbstractQueue** ()
- virtual bool **add** (const E &value) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.

- virtual bool **addAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

Adds all the elements of a collection to the queue.

- virtual E **remove** () throw (decaf::lang::exceptions::NoSuchElementException)

Retrieves and removes the head of this queue.

- virtual E **element** () const throw (decaf::lang::exceptions::NoSuchElementException)

Retrieves, but does not remove, the head of this queue.

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)

Removes all elements of the queue.

6.4.1 Detailed Description

template<typename E> class decaf::util::AbstractQueue< E >

This class provides skeletal implementations of some **Queue** (p. 2948) operations. Methods **add**, **remove**, and **element** are based on **offer**, **poll**, and **peek**, respectively.

A **Queue** (p. 2948) implementation that extends this class must minimally define a method **Queue** (p. 2948). **offer(E)** which does not permit insertion of null elements, along with methods **Queue** (p. 2948). **peek()** (p. 2950), **Queue.poll()** (p. 2950), **Collection.size()** (p. 1106), and a **Collection.iterator()** (p. 2012) supporting **Iterator.remove()** (p. 2014). Typically, additional methods will be overridden as well. If these requirements cannot be met, consider instead subclassing **AbstractCollection** (p. 143).

Since

1.0

6.4.2 Constructor & Destructor Documentation

6.4.2.1 **template<typename E > decaf::util::AbstractQueue< E >::AbstractQueue ()** [inline]

6.4.2.2 **template<typename E > virtual decaf::util::AbstractQueue< E >::~~AbstractQueue ()** [inline, virtual]

6.4.3 Member Function Documentation

6.4.3.1 **template<typename E > virtual bool decaf::util::AbstractQueue< E >::add (const E & *value*)** throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException) [inline, virtual]

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an **IllegalStateException** if no space is currently available.

This implementation returns true if **offer** succeeds, else throws an **IllegalStateException**.

Parameters

value - the element to offer to the **Queue** (p. 2948).

Returns

true if the add succeeds.

Exceptions

IllegalArgumentException if the element cannot be added.

Implements **decaf::util::Collection< E >** (p. 1099).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 2835).

References **decaf::util::Queue< E >::offer()**.

```

6.4.3.2  template<typename E > virtual bool decaf::util::AbstractQueue<
            E >::addAll ( const Collection< E > & collection )
            throw ( lang::exceptions::UnsupportedOperationException,
                    lang::exceptions::IllegalArgumentException,
                    lang::exceptions::IllegalStateException ) [inline,
                    virtual]

```

Adds all the elements of a collection to the queue.

If the collection is the queue itself, then an `IllegalArgumentException` will be thrown out. If during the process, some runtime exception is thrown out, then part of the elements in the collection that have successfully added will remain in the queue.

The result of the method is undefined if the collection is modified during the process of the method.

Parameters

collection - the collection to be added to the queue.

Returns

true if the operation succeeds.

Exceptions

IllegalArgumentException If the collection to be added to the queue is the queue itself.

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 147).

```

6.4.3.3  template<typename E > virtual void decaf::util::AbstractQueue< E >::clear
            ( ) throw ( lang::exceptions::UnsupportedOperationException ) [inline,
            virtual]

```

Removes all elements of the queue.

This implementation repeatedly invokes `poll` until it returns the empty marker.

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 147).

Reimplemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3480), and `decaf::util::PriorityQueue< E >` (p. 2836).

References `decaf::util::AbstractCollection< E >::isEmpty()`, and `decaf::util::Queue< E >::poll()`.

```

6.4.3.4  template<typename E > virtual E decaf::util::AbstractQueue< E >::element
            ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )
            [inline, virtual]

```

Retrieves, but does not remove, the head of this queue.

This method differs from `peek` only in that it throws an exception if this queue is empty.

This implementation returns the result of `peek` unless the queue is empty.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException if the queue is empty.

Implements **decaf::util::Queue< E >** (p. 2949).

References **decaf::util::Queue< E >::peek()**.

6.4.3.5 `template<typename E> virtual E decaf::util::AbstractQueue< E >::remove
() throw (decaf::lang::exceptions::NoSuchElementException) [inline,
virtual]`

Retrieves and removes the head of this queue.

This method differs from `poll` only in that it throws an exception if this queue is empty.

This implementation returns the result of `poll` unless the queue is empty.

Returns

a copy of the element in the head of the queue.

Exceptions

NoSuchElementException if the queue is empty.

Implements **decaf::util::Queue< E >** (p. 2950).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 2838).

References **decaf::util::Queue< E >::poll()**.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractQueue.h`

6.5 decaf::util::AbstractSequentialList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 2190) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).

```
#include <src/main/decaf/util/AbstractSequentialList.h>
```

Inheritance diagram for **decaf::util::AbstractSequentialList< E >**:

Public Member Functions

- `virtual ~AbstractSequentialList()`

6.5.1 Detailed Description

template<typename E> class decaf::util::AbstractSequentialList< E >

This class provides a skeletal implementation of the **List** (p. 2190) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list). For random access data (such as an array), **AbstractList** (p. 156) should be used in preference to this class.

This class is the opposite of the **AbstractList** (p. 156) class in the sense that it implements the "random access" methods (get(int index), set(int index, E element), add(int index, E element) and remove(int index)) on top of the list's list iterator, instead of the other way around.

To implement a list the programmer needs only to extend this class and provide implementations for the listIterator and size methods. For an unmodifiable list, the programmer need only implement the list iterator's hasNext, next, hasPrevious, previous and index methods.

For a modifiable list the programmer should additionally implement the list iterator's set method. For a variable-size list the programmer should additionally implement the list iterator's remove and add methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 1097) interface specification.

Since

1.0

6.5.2 Constructor & Destructor Documentation

6.5.2.1 template<typename E > virtual decaf::util::AbstractSequentialList< E >::~~AbstractSequentialList () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractSequentialList.h**

6.6 decaf::util::AbstractSet< E > Class Template Reference

This class provides a skeletal implementation of the **Set** (p. 3220) interface to minimize the effort required to implement this interface.

#include <src/main/decaf/util/AbstractSet.h>

Inheritance diagram for decaf::util::AbstractSet< E >:

Public Member Functions

- virtual ~**AbstractSet** ()
- virtual bool **removeAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

Removes from this set all of its elements that are contained in the specified collection (optional operation).

6.6.1 Detailed Description

template<typename E> class decaf::util::AbstractSet< E >

This class provides a skeletal implementation of the **Set** (p. 3220) interface to minimize the effort required to implement this interface. The process of implementing a set by extending this class is identical to that of implementing a **Collection** (p.1097) by extending **AbstractCollection** (p. 143), except that all of the methods and constructors in subclasses of this class must obey the additional constraints imposed by the **Set** (p. 3220) interface (for instance, the add method must not permit addition of multiple instances of an object to a set).

Since

1.0

6.6.2 Constructor & Destructor Documentation

6.6.2.1 **template<typename E> virtual decaf::util::AbstractSet< E >::~AbstractSet**
() [inline, virtual]

6.6.3 Member Function Documentation

6.6.3.1 **template<typename E> virtual bool decaf::util::AbstractSet<**
E >::removeAll (const Collection< E > & collection)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException) [inline, virtual]

Removes from this set all of its elements that are contained in the specified collection (optional operation).

If the specified collection is also a set, this operation effectively modifies this set so that its value is the asymmetric set difference of the two sets.

This implementation determines which is the smaller of this set and the specified collection, by invoking the size method on each. If this set has fewer elements, then the implementation iterates over this set, checking each element returned by the iterator in turn to see if it is contained in the specified collection. If it is so contained, it is removed from this set with the iterator's remove method. If the specified collection has fewer elements, then the implementation iterates over the specified collection, removing from this set each element returned by the iterator, using this set's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method.

Parameters

collection - The **Collection** (p. 1097) whose elements are to be retained

Returns

true if the collection changed as a result of this call

Exceptions

UnsupportedOperationException

IllegalArgumentException

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 152).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractSet.h`

6.7 activemq::transport::AbstractTransportFactory Class Reference

Abstract implementation of the `TransportFactory` (p. 3634) interface, providing the base functionality that's common to most of the `TransportFactory` (p. 3634) instances.

`#include <src/main/activemq/transport/AbstractTransportFactory.h>`

Inheritance diagram for `activemq::transport::AbstractTransportFactory`:

Public Member Functions

- `virtual ~AbstractTransportFactory ()`

Protected Member Functions

- `virtual Pointer< wireformat::WireFormat > createWireFormat (const decaf::util::Properties &properties) throw (decaf::lang::exceptions::NoSuchElementException)`

*Creates the WireFormat that is configured for this **Transport** (p. 3629) and returns it.*

6.7.1 Detailed Description

Abstract implementation of the `TransportFactory` (p. 3634) interface, providing the base functionality that's common to most of the `TransportFactory` (p. 3634) instances.

Since

3.0

6.7.2 Constructor & Destructor Documentation

- 6.7.2.1** virtual
 activemq::transport::AbstractTransportFactory::~~AbstractTransportFactory
 () [inline, virtual]

6.7.3 Member Function Documentation

- 6.7.3.1** virtual Pointer<wireformat::WireFormat> ac-
 tivemq::transport::AbstractTransportFactory::createWireFormat
 (const decaf::util::Properties & *properties*) throw (
 decaf::lang::exceptions::NoSuchElementException) [protected, virtual]

Creates the WireFormat that is configured for this **Transport** (p. 3629) and returns it.

The default WireFormat is Openwire.

Parameters

properties The properties that were configured on the URI.

Returns

a pointer to a WireFormat instance that the caller then owns.

Exceptions

NoSuchElementException if the configured WireFormat is not found.

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**AbstractTransportFactory.h**

6.8 activemq::core::ActiveMQAckHandler Class Reference

Interface class that is used to give CMS Messages an interface to Ack themselves with.

```
#include <src/main/activemq/core/ActiveMQAckHandler.h>
```

Public Member Functions

- virtual ~**ActiveMQAckHandler** ()
- virtual void **acknowledgeMessage** (const **commands::Message** *message)=0 throw (
 cms::CMSException)

Method called to acknowledge the message passed.

6.8.1 Detailed Description

Interface class that is used to give CMS Messages an interface to Ack themselves with.

Since

2.0

6.8.2 Constructor & Destructor Documentation

- 6.8.2.1 `virtual activemq::core::ActiveMQAckHandler::~~ActiveMQAckHandler ()`
[inline, virtual]

6.8.3 Member Function Documentation

- 6.8.3.1 `virtual void activemq::core::ActiveMQAckHandler::acknowledgeMessage (const commands::Message * message) throw (cms::CMSException)`
[pure virtual]

Method called to acknowledge the message passed.

Parameters

message Message to Acknowledge

Exceptions

CMSException

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQAckHandler.h`

6.9 activemq::commands::ActiveMQBlobMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQBlobMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQBlobMessage`:

Public Member Functions

- `ActiveMQBlobMessage ()`
- `virtual ~ActiveMQBlobMessage ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual ActiveMQBlobMessage * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- `virtual void copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this objects members, overwriting any existing data.
- `virtual std::string toString () const`
Returns a string containing the information for this DataStructure (p.1553) such as its type and value of its elements.

- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- std::string **getRemoteBlobUrl** () const
Get the Remote URL of the Blob.
- void **setRemoteBlobUrl** (const std::string &remoteURL)
Set the Remote URL of the Blob.
- std::string **getMimeType** () const
Get the Mime Type of the Blob.
- void **setMimeType** (const std::string &mimeType)
Set the Mime Type of the Blob.
- std::string **getName** () const
Gets the Name of the Blob.
- void **setName** (const std::string &name)
Sets the Name of the Blob.
- bool **isDeletedByBroker** () const
Gets if this Blob is deleted by the Broker.
- void **setDeletedByBroker** (bool value)
Sets the Deleted By Broker flag.

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQBLOBMESSAGE** = 29
- static const std::string **BINARY _MIME _TYPE**

6.9.1 Constructor & Destructor Documentation

6.9.1.1 `activemq::commands::ActiveMQBlobMessage::ActiveMQBlobMessage ()`

6.9.1.2 `virtual
activemq::commands::ActiveMQBlobMessage::~~ActiveMQBlobMessage ()` [inline, virtual]

6.9.2 Member Function Documentation

6.9.2.1 `virtual cms::Message* activemq::commands::ActiveMQBlobMessage::clone ()` const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.9.2.2 `virtual ActiveMQBlobMessage* activemq::commands::ActiveMQBlobMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2363).

6.9.2.3 `virtual void activemq::commands::ActiveMQBlobMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2363).

6.9.2.4 `virtual bool activemq::commands::ActiveMQBlobMessage::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 383).

6.9.2.5 `virtual unsigned char activemq::commands::ActiveMQBlobMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1553) type copy.

Reimplemented from `activemq::commands::Message` (p. 2365).

6.9.2.6 `std::string activemq::commands::ActiveMQBlobMessage::getMimeType () const [inline]`

Get the Mime Type of the Blob.

Returns

string holding the MIME Type.

6.9.2.7 `std::string activemq::commands::ActiveMQBlobMessage::getName () const [inline]`

Gets the Name of the Blob.

Returns

string name of the Blob.

6.9.2.8 `std::string activemq::commands::ActiveMQBlobMessage::getRemoteBlobUrl () const [inline]`

Get the Remote URL of the Blob.

Returns

string from of the Remote Blob URL.

6.9.2.9 `bool activemq::commands::ActiveMQBlobMessage::isDeletedByBroker () const [inline]`

Gets if this Blob is deleted by the Broker.

Returns

true if the Blob is deleted by the Broker.

6.9.2.10 `void activemq::commands::ActiveMQBlobMessage::setDeletedByBroker (bool value) [inline]`

Sets the Deleted By Broker flag.

Parameters

value - set the Delete by broker flag to value.

6.9.2.11 `void activemq::commands::ActiveMQBlobMessage::setMimeType (const std::string & mimeType) [inline]`

Set the Mime Type of the Blob.

Parameters

mimeType - String holding the MIME Type.

6.9.2.12 `void activemq::commands::ActiveMQBlobMessage::setName (const std::string & name) [inline]`

Sets the Name of the Blob.

Parameters

name - Name of the Blob.

6.9.2.13 `void activemq::commands::ActiveMQBlobMessage::setRemoteBlobUrl (const std::string & remoteURL) [inline]`

Set the Remote URL of the Blob.

Parameters

remoteURL - String form of the Remote URL.

6.9.2.14 `virtual std::string activemq::commands::ActiveMQBlobMessage::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p.2372).

6.9.3 Field Documentation

6.9.3.1 `const std::string activemq::commands::ActiveMQBlobMessage::BINARY_MIME_TYPE [static]`

6.9.3.2 `const unsigned char activemq::commands::ActiveMQBlobMessage::ID_ACTIVEMQBLOBMESSAGE = 29 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQBlobMessage.h`

6.10 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 171).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.10.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 171).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.10.2 Constructor & Destructor Documentation

6.10.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

6.10.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

6.10.3 Member Function Documentation

6.10.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.10.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.10.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

6.10.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

6.10.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2530).

6.10.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2531).

6.10.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2531).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h`

6.11 `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 174).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.11.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.174).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.11.2 Constructor & Destructor Documentation

6.11.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller () [inline]`

6.11.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller () [inline, virtual]`

6.11.3 Member Function Documentation

6.11.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.11.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.11.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

6.11.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

6.11.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2542).

6.11.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543).

```
6.11.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h`

6.12 `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 178).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.12.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.178).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.12.2 Constructor & Destructor Documentation

6.12.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

6.12.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

6.12.3 Member Function Documentation

6.12.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.12.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.12.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

6.12.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

6.12.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2538).

6.12.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539).

```
6.12.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h`

6.13 `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 182).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.13.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.182).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.13.2 Constructor & Destructor Documentation

6.13.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller () [inline]`

6.13.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller () [inline, virtual]`

6.13.3 Member Function Documentation

6.13.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.13.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.13.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

6.13.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

6.13.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2534).

6.13.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535).

```
6.13.3.7 virtual void ac-
      tivemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightUnmars
      ( OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataInputStream * dataIn,
      utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h`

6.14 `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 186).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.14.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.186).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.14.2 Constructor & Destructor Documentation

6.14.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller () [inline]`

6.14.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller () [inline, virtual]`

6.14.3 Member Function Documentation

6.14.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.14.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.14.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2524).

6.14.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2525).

6.14.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

6.14.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

```
6.14.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2527).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h`

6.15 `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 190).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller`:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.15.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.190).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.15.2 Constructor & Destructor Documentation

6.15.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller () [inline]`

6.15.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller () [inline, virtual]`

6.15.3 Member Function Documentation

6.15.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.15.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.15.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

6.15.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

6.15.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2546).

6.15.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2547).

```
6.15.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2547).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarshaller.h`

6.16 `activemq::commands::ActiveMQBytesMessage` Class Reference

```
#include <src/main/activemq/commands/ActiveMQBytesMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQBytesMessage`:

Public Member Functions

- **ActiveMQBytesMessage** ()
- virtual **~ActiveMQBytesMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQBytesMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual **cms::BytesMessage * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual void **clearBody** () throw (cms::CMSEException)
Clears out the body of the message.
- virtual void **onSend** ()
Store the Data that was written to the stream before a send.
- virtual void **setBodyBytes** (const unsigned char *buffer, int numBytes) throw (cms::MessageNotWriteableException, cms::CMSEException)
sets the bytes given to the message body.
- virtual unsigned char * **getBodyBytes** () const throw (cms::MessageNotReadableException, cms::CMSEException)
Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.
- virtual int **getBodyLength** () const throw (cms::MessageNotReadableException, cms::CMSEException)
Returns the number of bytes contained in the body of this message.
- virtual void **reset** () throw (cms::MessageFormatException, cms::CMSEException)
Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Boolean from the Bytes message stream.

- virtual void **writeBoolean** (bool value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a boolean to the bytes message stream as a 1-byte value.
- virtual unsigned char **readByte** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Byte from the Bytes message stream.
- virtual void **writeByte** (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte to the bytes message stream as a 1-byte value.
- virtual int **readBytes** (std::vector< unsigned char > &value) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a byte array from the bytes message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.
- virtual int **readBytes** (unsigned char *buffer, int length) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a portion of the bytes message stream.
- virtual void **writeBytes** (const unsigned char *value, int offset, int length) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a portion of a byte array to the bytes message stream.
- virtual char **readChar** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Char from the Bytes message stream.
- virtual void **writeChar** (char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a char to the bytes message stream as a 1-byte value.
- virtual float **readFloat** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 32 bit float from the Bytes message stream.
- virtual void **writeFloat** (float value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a float to the bytes message stream as a 4 byte value.
- virtual double **readDouble** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 64 bit double from the Bytes message stream.

- virtual void **writeDouble** (double value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a double to the bytes message stream as a 8 byte value.
- virtual short **readShort** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 16 bit signed short from the Bytes message stream.
- virtual void **writeShort** (short value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a signed short to the bytes message stream as a 2 byte value.
- virtual unsigned short **readUnsignedShort** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 16 bit unsigned short from the Bytes message stream.
- virtual void **writeUnsignedShort** (unsigned short value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a unsigned short to the bytes message stream as a 2 byte value.
- virtual int **readInt** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 32 bit signed integer from the Bytes message stream.
- virtual void **writeInt** (int value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a signed int to the bytes message stream as a 4 byte value.
- virtual long long **readLong** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 64 bit long from the Bytes message stream.
- virtual void **writeLong** (long long value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a long long to the bytes message stream as a 8 byte value.
- virtual std::string **readString** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads an ASCII String from the Bytes message stream.
- virtual void **writeString** (const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes an ASCII String to the Bytes message stream.
- virtual std::string **readUTF** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads an UTF String from the BytesMessage stream.
- virtual void **writeUTF** (const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes an UTF String to the BytesMessage stream.

Static Public Attributes

- static const unsigned char `ID_ACTIVEMQBYTESMESSAGE` = 24

6.16.1 Constructor & Destructor Documentation

6.16.1.1 `activemq::commands::ActiveMQBytesMessage::ActiveMQBytesMessage ()`

6.16.1.2 `virtual
activemq::commands::ActiveMQBytesMessage::~~ActiveMQBytesMessage
() [virtual]`

6.16.2 Member Function Documentation

6.16.2.1 `virtual void activemq::commands::ActiveMQBytesMessage::clearBody () throw (cms::CMSException) [virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 383).

6.16.2.2 `virtual cms::BytesMessage* activemq::commands::ActiveMQBytesMessage::clone ()
const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.16.2.3 `virtual ActiveMQBytesMessage* activemq::commands::ActiveMQBytesMessage::cloneDataStructure ()
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2363).

6.16.2.4 `virtual void activemq::commands::ActiveMQBytesMessage::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2363).

6.16.2.5 virtual bool activemq::commands::ActiveMQBytesMessage::equals (const DataStructure * value) const [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Parameters

value The **Command** (p. 1107) to compare to this one.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 383).

6.16.2.6 virtual unsigned char* activemq::commands::ActiveMQBytesMessage::getBodyBytes () const throw (cms::MessageNotReadableException, cms::CMSEException) [virtual]

Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.

Call `getBodyLength` to determine the number of bytes to expect.

Returns

const pointer to a byte buffer

Exceptions

CMSEException - If an internal error occurs.

MessageNotReadableException - If the message is in Write Only Mode.

6.16.2.7 virtual int activemq::commands::ActiveMQBytesMessage::getBodyLength () const throw (cms::MessageNotReadableException, cms::CMSEException) [virtual]

Returns the number of bytes contained in the body of this message.

Returns

number of bytes.

Exceptions

CMSException - If an internal error occurs.

MessageNotReadableException - If the message is in Write Only Mode.

6.16.2.8 `virtual unsigned char activemq::commands::ActiveMQBytesMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Reimplemented from **activemq::commands::Message** (p. 2365).

6.16.2.9 `virtual void activemq::commands::ActiveMQBytesMessage::onSend () [virtual]`

Store the Data that was written to the stream before a send.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>** (p. 390).

6.16.2.10 `virtual bool activemq::commands::ActiveMQBytesMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a Boolean from the Bytes message stream.

Returns

boolean value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.11 `virtual unsigned char activemq::commands::ActiveMQBytesMessage::readByte () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a Byte from the Bytes message stream.

Returns

unsigned char value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.12 `virtual int activemq::commands::ActiveMQBytesMessage::readBytes
(std::vector< unsigned char > & value) const throw (
cms::MessageEOFException, cms::MessageNotReadableException,
cms::CMSException) [virtual]`

Reads a byte array from the bytes message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

value buffer to place data in

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.13 `virtual int activemq::commands::ActiveMQBytesMessage::readBytes
(unsigned char * buffer, int length) const throw (
cms::MessageEOFException, cms::MessageNotReadableException,
cms::CMSException) [virtual]`

Reads a portion of the bytes message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an IndexOutOfBoundsException is thrown. No bytes will be read from the stream for this exception case.

Parameters

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.14 `virtual char activemq::commands::ActiveMQBytesMessage::readChar
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a Char from the Bytes message stream.

Returns

char value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.15 `virtual double ac-
tivismq::commands::ActiveMQBytesMessage::readDouble
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a 64 bit double from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.16 virtual float activemq::commands::ActiveMQBytesMessage::readFloat
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 32 bit float from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.17 virtual int activemq::commands::ActiveMQBytesMessage::readInt
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 32 bit signed integer from the Bytes message stream.

Returns

int value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.18 virtual long long ac-
tivemq::commands::ActiveMQBytesMessage::readLong
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 64 bit long from the Bytes message stream.

Returns

long long value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.19 virtual short activemq::commands::ActiveMQBytesMessage::readShort
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSEException) [virtual]

Reads a 16 bit signed short from the Bytes message stream.

Returns

short value from stream

Exceptions

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.20 virtual std::string ac-
tivismq::commands::ActiveMQBytesMessage::readString
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSEException) [virtual]

Reads an ASCII String from the Bytes message stream.

Returns

String from stream

Exceptions

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.21 virtual unsigned short ac-
tivismq::commands::ActiveMQBytesMessage::readUnsignedShort
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSEException) [virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

Returns

unsigned short value from stream

Exceptions

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.22 virtual std::string activemq::commands::ActiveMQBytesMessage::readUTF () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads an UTF String from the BytesMessage stream.

Returns

String from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.2.23 virtual void activemq::commands::ActiveMQBytesMessage::reset () throw (cms::MessageFormatException, cms::CMSException) [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions

CMSException - If the provider fails to perform the reset operation.

MessageFormatException - If the **Message** (p. 2358) has an invalid format.

6.16.2.24 virtual void activemq::commands::ActiveMQBytesMessage::setBodyBytes (const unsigned char * *buffer*, int *numBytes*) throw (cms::MessageNotWritableException, cms::CMSException) [virtual]

sets the bytes given to the message body.

Parameters

buffer Byte Buffer to copy

numBytes Number of bytes in Buffer to copy

Exceptions

CMSException - If an internal error occurs.

MessageNotWritableException - if in Read Only Mode.

6.16.2.25 virtual std::string activemq::commands::ActiveMQBytesMessage::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2372).

6.16.2.26 `virtual void activemq::commands::ActiveMQBytesMessage::writeBoolean (bool value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a boolean to the bytes message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

value boolean to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.27 `virtual void activemq::commands::ActiveMQBytesMessage::writeByte (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a byte to the bytes message stream as a 1-byte value.

Parameters

value byte to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.28 `virtual void activemq::commands::ActiveMQBytesMessage::writeBytes (const unsigned char * value, int offset, int length) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a portion of a byte array to the bytes message stream.

size as the number of bytes to write.

Parameters

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.29 virtual void activemq::commands::ActiveMQBytesMessage::writeBytes
(const std::vector< unsigned char > & *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

Parameters

value bytes to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.30 virtual void activemq::commands::ActiveMQBytesMessage::writeChar
(char *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a char to the bytes message stream as a 1-byte value.

Parameters

value char to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.31 virtual void activemq::commands::ActiveMQBytesMessage::writeDouble
(double *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a double to the bytes message stream as a 8 byte value.

Parameters

value double to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.32 virtual void activemq::commands::ActiveMQBytesMessage::writeFloat
(float *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a float to the bytes message stream as a 4 byte value.

Parameters

value float to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.33 virtual void activemq::commands::ActiveMQBytesMessage::writeInt
(int *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a signed int to the bytes message stream as a 4 byte value.

Parameters

value signed int to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.34 virtual void activemq::commands::ActiveMQBytesMessage::writeLong
(long long *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a long long to the bytes message stream as a 8 byte value.

Parameters

value signed long long to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.35 virtual void activemq::commands::ActiveMQBytesMessage::writeShort
(short *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a signed short to the bytes message stream as a 2 byte value.

Parameters

value signed short to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.36 virtual void activemq::commands::ActiveMQBytesMessage::writeString (const std::string & *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes an ASCII String to the Bytes message stream.

Parameters

value String to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.37 virtual void activemq::commands::ActiveMQBytesMessage::writeUnsignedShort (unsigned short *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters

value unsigned short to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.16.2.38 virtual void activemq::commands::ActiveMQBytesMessage::writeUTF (const std::string & *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes an UTF String to the BytesMessage stream.

Parameters

value String to write to the stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.16.3 Field Documentation

6.16.3.1 const unsigned char activemq::commands::ActiveMQBytesMessage::ID_ - ACTIVEMQBYTESMESSAGE = 24 [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQBytesMessage.h`

6.17 `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBytesMessageMarshaller` (p. 210).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`:

Public Member Functions

- `ActiveMQBytesMessageMarshaller ()`
- `virtual ~ActiveMQBytesMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.17.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 210).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.17.2 Constructor & Destructor Documentation

6.17.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

6.17.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

6.17.3 Member Function Documentation

6.17.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.17.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.17.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseMarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

```
6.17.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseUnmar
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

```
6.17.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarsh
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2530).

6.18

activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller

Class Reference

215

```
6.17.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2531).

```
6.17.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightUnmar
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2531).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h

6.18 activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMess Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 213).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller`:

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.18.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.213).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.18.2 Constructor & Destructor Documentation

6.18.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller () [inline]`

6.18.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller () [inline, virtual]`

6.18.3 Member Function Documentation

6.18.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.18.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.18.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

6.18.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

6.18.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2542).

6.18.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

6.19

activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller

Class Reference

219

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2543).

6.18.3.7 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataInputStream * dataIn,  
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2543).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h

6.19 activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 217).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**:

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.19.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 217).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.19.2 Constructor & Destructor Documentation

6.19.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller () [inline]`

6.19.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller () [inline, virtual]`

6.19.3 Member Function Documentation

6.19.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.19.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.19.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

6.19.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

6.19.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2538).

6.19.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

6.20

activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller

Class Reference

223

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2539).

6.19.3.7 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataInputStream * dataIn,  
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2539).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h

6.20 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 221).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller**:

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.20.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 221).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.20.2 Constructor & Destructor Documentation

6.20.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller () [inline]`

6.20.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller () [inline, virtual]`

6.20.3 Member Function Documentation

6.20.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.20.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.20.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2524).

6.20.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2525).

6.20.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

6.20.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

6.21

activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller

Class Reference

227

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2526).

6.20.3.7 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataInputStream * dataIn,  
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2527).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h

6.21 activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 225).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**:

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.21.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 225).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.21.2 Constructor & Destructor Documentation

6.21.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller () [inline]`

6.21.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller () [inline, virtual]`

6.21.3 Member Function Documentation

6.21.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.21.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.21.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

6.21.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

6.21.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2546).

6.21.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

6.22

activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller

Class Reference

231

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2547).

6.21.3.7 virtual void ac-

```
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightUnmarshal(  
    ( OpenWireFormat * wireFormat, commands::DataStructure  
    * dataStructure, decaf::io::DataInputStream * dataIn,  
    utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2547).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMessageMarshaller.h

6.22 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 229).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**:

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.22.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 229).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.22.2 Constructor & Destructor Documentation

6.22.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller () [inline]`

6.22.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller () [inline, virtual]`

6.22.3 Member Function Documentation

6.22.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.22.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.22.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

6.22.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

6.22.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2534).

6.22.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535).

```
6.22.3.7 virtual void ac-
      tivemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightUnmar
      ( OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataInputStream * dataIn,
      utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h`

6.23 activemq::core::ActiveMQConnection Class Reference

Concrete connection used for all connectors to the ActiveMQ broker.

```
#include <src/main/activemq/core/ActiveMQConnection.h>
```

Inheritance diagram for `activemq::core::ActiveMQConnection`:

Public Member Functions

- **ActiveMQConnection** (const **Pointer**< **transport::Transport** > &transport, const **Pointer**< **decaf::util::Properties** > &properties)
Constructor.
- virtual ~**ActiveMQConnection** ()
- virtual void **removeSession** (**ActiveMQSession** *session) throw (cms::CMSException)
Removes the session resources for the given session instance.
- virtual void **addProducer** (**ActiveMQProducer** *producer) throw (cms::CMSException)
Adds an active Producer to the Set of known producers.
- virtual void **removeProducer** (const **Pointer**< **commands::ProducerId** > &producerId) throw (cms::CMSException)
Removes an active Producer to the Set of known producers.
- virtual void **addDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer, **Dispatcher** *dispatcher) throw (cms::CMSException)
Adds a dispatcher for a consumer.
- virtual void **removeDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer) throw (cms::CMSException)
Removes the dispatcher for a consumer.
- virtual void **sendPullRequest** (const **commands::ConsumerInfo** *consumer, long long timeout) throw (exceptions::ActiveMQException)
If supported sends a message pull request to the service provider asking for the delivery of a new message.
- bool **isClosed** () const
Checks if this connection has been closed.
- bool **isStarted** () const
Check if this connection has been started.
- bool **isTransportFailed** () const
Checks if the Connection's Transport has failed.
- virtual void **destroyDestination** (const **commands::ActiveMQDestination** *destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException)
Requests that the Broker removes the given Destination.
- virtual void **destroyDestination** (const **cms::Destination** *destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException)

Requests that the Broker removes the given Destination.

- virtual const **cms::ConnectionMetaData** * **getMetaData** () const throw (cms::CMSEException)

Gets the metadata for this connection.

- virtual **cms::Session** * **createSession** () throw (cms::CMSEException)

Creates a new Session to work for this Connection.

- virtual std::string **getClientID** () const

- virtual void **setClientID** (const std::string &clientID)

- virtual **cms::Session** * **createSession** (cms::Session::AcknowledgeMode ackMode) throw (cms::CMSEException)

Creates a new Session to work for this Connection using the specified acknowledgment mode.

- virtual void **close** () throw (cms::CMSEException)

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

- virtual void **start** () throw (cms::CMSEException)

Starts or (restarts) a connections delivery of incoming messages.

- virtual void **stop** () throw (cms::CMSEException)

Stop the flow of incoming messages.

- virtual **cms::ExceptionListener** * **getExceptionListener** () const

Gets the registered Exception Listener for this connection.

- virtual void **setExceptionListener** (cms::ExceptionListener *listener)

Sets the registered Exception Listener for this connection.

- void **setUsername** (const std::string &username)

Sets the username that should be used when creating a new connection.

- const std::string & **getUsername** () const

Gets the username that this factory will use when creating a new connection instance.

- void **setPassword** (const std::string &password)

Sets the password that should be used when creating a new connection.

- const std::string & **getPassword** () const

Gets the password that this factory will use when creating a new connection instance.

- void **setDefaultClientId** (const std::string &clientId)

Sets the Client Id.

- void **setBrokerURL** (const std::string &brokerURL)
Sets the Broker URL that should be used when creating a new connection instance.
- const std::string & **getBrokerURL** () const
Gets the Broker URL that this factory will use when creating a new connection instance.
- void **setPrefetchPolicy** (**PrefetchPolicy** *policy)
*Sets the **PrefetchPolicy** (p. 2783) instance that this factory should use when it creates new Connection instances.*
- **PrefetchPolicy** * **getPrefetchPolicy** () const
*Gets the pointer to the current **PrefetchPolicy** (p. 2783) that is in use by this ConnectionFactory.*
- void **setRedeliveryPolicy** (**RedeliveryPolicy** *policy)
*Sets the **RedeliveryPolicy** (p. 2972) instance that this factory should use when it creates new Connection instances.*
- **RedeliveryPolicy** * **getRedeliveryPolicy** () const
*Gets the pointer to the current **RedeliveryPolicy** (p. 2972) that is in use by this ConnectionFactory.*
- bool **isDispatchAsync** () const
- void **setDispatchAsync** (bool value)
Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.
- bool **isAlwaysSyncSend** () const
Gets if the Connection should always send things Synchronously.
- void **setAlwaysSyncSend** (bool value)
Sets if the Connection should always send things Synchronously.
- bool **isUseAsyncSend** () const
Gets if the useAsyncSend option is set.
- void **setUseAsyncSend** (bool value)
Sets the useAsyncSend option.
- bool **isUseCompression** () const
Gets if the Connection is configured for Message body compression.
- void **setUseCompression** (bool value)
Sets whether Message body compression is enabled.
- unsigned int **getSendTimeout** () const
Gets the assigned send timeout for this Connector.
- void **setSendTimeout** (unsigned int timeout)
Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

- unsigned int **getCloseTimeout** () const
Gets the assigned close timeout for this Connector.
- void **setCloseTimeout** (unsigned int timeout)
Sets the close timeout to use when sending the disconnect request.
- unsigned int **getProducerWindowSize** () const
Gets the configured producer window size for Producers that are created from this connector.
- void **setProducerWindowSize** (unsigned int windowSize)
Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.
- long long **getNextSessionId** ()
Get the Next available Session Id.
- long long **getNextTempDestinationId** ()
Get the Next Temporary Destination Id.
- long long **getNextLocalTransactionId** ()
Get the Next Temporary Destination Id.
- void **addTransportListener** (transport::TransportListener *transportListener)
Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the Connection class.
- void **removeTransportListener** (transport::TransportListener *transportListener)
Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events.
- virtual void **onCommand** (const Pointer< commands::Command > &command)
Event handler for the receipt of a non-response command from the transport.
- virtual void **onException** (const decaf::lang::Exception &ex)
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.
- const commands::ConnectionInfo & **getConnectionInfo** () const throw (exceptions::ActiveMQException)
Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.
- const commands::ConnectionId & **getConnectionId** () const throw (exceptions::ActiveMQException)

Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.

- **transport::Transport & getTransport () const**

Gets a reference to this object's Transport instance.

- **void oneway (Pointer< commands::Command > command) throw (activemq::exceptions::ActiveMQException)**

Sends a oneway message.

- **void syncRequest (Pointer< commands::Command > command, unsigned int timeout=0) throw (activemq::exceptions::ActiveMQException)**

Sends a synchronous request and returns the response from the broker.

- **virtual void fire (const exceptions::ActiveMQException &ex)**

Notify the exception listener.

- **void setTransportInterruptionProcessingComplete ()**

Indicates that a Connection resource that is processing the transportInterrupted event has completed.

6.23.1 Detailed Description

Concrete connection used for all connectors to the ActiveMQ broker.

Since

2.0

6.23.2 Constructor & Destructor Documentation

- #### 6.23.2.1 **activemq::core::ActiveMQConnection::ActiveMQConnection (const Pointer< transport::Transport > & transport, const Pointer< decaf::util::Properties > & properties)**

Constructor.

Parameters

transport The Transport requested for this connection to the Broker.

properties The Properties that were defined for this connection

6.23.2.2 `virtual activemq::core::ActiveMQConnection::~~ActiveMQConnection ()`
[virtual]

6.23.3 Member Function Documentation

6.23.3.1 `virtual void activemq::core::ActiveMQConnection::addDispatcher (`
`const Pointer< commands::ConsumerId > & consumer, Dispatcher *`
`dispatcher) throw (cms::CMSException)` [virtual]

Adds a dispatcher for a consumer.

Parameters

consumer - The consumer for which to register a dispatcher.

dispatcher - The dispatcher to handle incoming messages for the consumer.

6.23.3.2 `virtual void activemq::core::ActiveMQConnection::addProducer (`
`ActiveMQProducer * producer) throw (cms::CMSException)`
[virtual]

Adds an active Producer to the Set of known producers.

Parameters

producer - The Producer to add from the the known set.

6.23.3.3 `void activemq::core::ActiveMQConnection::addTransportListener (`
`transport::TransportListener * transportListener)`

Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the Connection class.

Client's should ensure that the registered listener does not block or take a long amount of time to execute in order to not degrade performance of this Connection.

Parameters

transportListener The TransportListener instance to add to this Connection's set of listeners to notify of Transport events.

6.23.3.4 `virtual void activemq::core::ActiveMQConnection::close () throw (`
`cms::CMSException)` [virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions

CMSException

6.23.3.5 `virtual cms::Session* activemq::core::ActiveMQConnection::createSession (cms::Session::AcknowledgeMode ackMode) throw (cms::CMSException) [virtual]`

Creates a new Session to work for this Connection using the specified acknowledgment mode.

Parameters

ackMode the Acknowledgment Mode to use.

Exceptions

CMSException

6.23.3.6 `virtual cms::Session* activemq::core::ActiveMQConnection::createSession () throw (cms::CMSException) [virtual]`

Creates a new Session to work for this Connection.

Exceptions

CMSException

6.23.3.7 `virtual void activemq::core::ActiveMQConnection::destroyDestination (const commands::ActiveMQDestination * destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException) [virtual]`

Requests that the Broker removes the given Destination.

Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

Parameters

destination The Destination the Broker will be requested to remove.

Exceptions

NullPointerException If the passed Destination is Null

IllegalStateException If the connection is closed.

UnsupportedOperationException If the wire format in use does not support this operation.

ActiveMQException If any other error occurs during the attempt to destroy the destination.

6.23.3.8 `virtual void activemq::core::ActiveMQConnection::destroyDestination (const cms::Destination * destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException) [virtual]`

Requests that the Broker removes the given Destination.

Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

Parameters

destination The CMS Destination the Broker will be requested to remove.

Exceptions

NullPointerException If the passed Destination is Null

IllegalStateException If the connection is closed.

UnsupportedOperationException If the wire format in use does not support this operation.

ActiveMQException If any other error occurs during the attempt to destroy the destination.

6.23.3.9 `virtual void activemq::core::ActiveMQConnection::fire (const exceptions::ActiveMQException & ex) [virtual]`

Notify the exception listener.

Parameters

ex the exception to fire

6.23.3.10 `const std::string& activemq::core::ActiveMQConnection::getBrokerURL () const`

Gets the Broker URL that this factory will use when creating a new connection instance.

Returns

brokerURL string

6.23.3.11 `virtual std::string activemq::core::ActiveMQConnection::getClientID () const [virtual]`

6.23.3.12 `unsigned int activemq::core::ActiveMQConnection::getCloseTimeout () const`

Gets the assigned close timeout for this Connector.

Returns

the close timeout configured in the connection uri

6.23.3.13 `const commands::ConnectionId& activemq::core::ActiveMQConnection::getConnectionId () const throw (exceptions::ActiveMQException)`

Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.

6.23.3.14 `const commands::ConnectionInfo& activemq::core::ActiveMQConnection::getConnectionInfo () const throw (exceptions::ActiveMQException)`

Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.

6.23.3.15 `virtual cms::ExceptionListener* activemq::core::ActiveMQConnection::getExceptionListener () const [virtual]`

Gets the registered Exception Listener for this connection.

Returns

pointer to an exception listener or NULL

6.23.3.16 `virtual const cms::ConnectionMetaData* activemq::core::ActiveMQConnection::getMetaData () const throw (cms::CMSException) [inline, virtual]`

Gets the metadata for this connection.

Returns

the connection MetaData pointer (caller does not own it).

Exceptions

CMSException if the provider fails to get the connection metadata for this connection.

See also

ConnectionMetaData

Since

2.0

6.23.3.17 `long long activemq::core::ActiveMQConnection::getNextLocalTransactionId ()`

Get the Next Temporary Destination Id.

Returns

the next id in the sequence.

6.23.3.18 `long long activemq::core::ActiveMQConnection::getNextSessionId ()`

Get the Next available Session Id.

Returns

the next id in the sequence.

6.23.3.19 `long long activemq::core::ActiveMQConnection::getNextTempDestinationId ()`

Get the Next Temporary Destination Id.

Returns

the next id in the sequence.

6.23.3.20 `const std::string& activemq::core::ActiveMQConnection::getPassword () const`

Gets the password that this factory will use when creating a new connection instance.

Returns

password string, "" for default credentials

6.23.3.21 `PrefetchPolicy* activemq::core::ActiveMQConnection::getPrefetchPolicy () const`

Gets the pointer to the current **PrefetchPolicy** (p. 2783) that is in use by this ConnectionFactory.

Returns

a pointer to this objects **PrefetchPolicy** (p. 2783).

6.23.3.22 `unsigned int activemq::core::ActiveMQConnection::getProducerWindowSize () const`

Gets the configured producer window size for Producers that are created from this connector. This only applies if there is no send timeout and the producer is able to send asynchronously.

Returns

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

6.23.3.23 `RedeliveryPolicy* activemq::core::ActiveMQConnection::getRedeliveryPolicy () const`

Gets the pointer to the current **RedeliveryPolicy** (p. 2972) that is in use by this ConnectionFactory.

Returns

a pointer to this objects **RedeliveryPolicy** (p. 2972).

6.23.3.24 `unsigned int activemq::core::ActiveMQConnection::getSendTimeout () const`

Gets the assigned send timeout for this Connector.

Returns

the send timeout configured in the connection uri

6.23.3.25 `transport::Transport& activemq::core::ActiveMQConnection::getTransport () const`

Gets a reference to this object's Transport instance.

Returns

a reference to the Transport that is in use by this Connection.

6.23.3.26 `const std::string& activemq::core::ActiveMQConnection::getUsername () const`

Gets the username that this factory will use when creating a new connection instance.

Returns

username string, "" for default credentials

6.23.3.27 bool activemq::core::ActiveMQConnection::isAlwaysSyncSend () const

Gets if the Connection should always send things Synchronously.

Returns

true if sends should always be Synchronous.

6.23.3.28 bool activemq::core::ActiveMQConnection::isClosed () const [inline]

Checks if this connection has been closed.

Returns

true if the connection is closed

6.23.3.29 bool activemq::core::ActiveMQConnection::isDispatchAsync () const**Returns**

The value of the dispatch asynchronously option sent to the broker.

6.23.3.30 bool activemq::core::ActiveMQConnection::isStarted () const [inline]

Check if this connection has been started.

Returns

true if the start method has been called.

6.23.3.31 bool activemq::core::ActiveMQConnection::isTransportFailed () const [inline]

Checks if the Connection's Transport has failed.

Returns

true if the Connection's Transport has failed.

6.23.3.32 bool activemq::core::ActiveMQConnection::isUseAsyncSend () const

Gets if the useAsyncSend option is set.

Returns

true if on false if not.

6.23.3.33 `bool activemq::core::ActiveMQConnection::isUseCompression () const`

Gets if the Connection is configured for Message body compression.

Returns

if the Message body will be Compressed or not.

6.23.3.34 `virtual void activemq::core::ActiveMQConnection::onCommand (const Pointer< commands::Command > & command) [virtual]`

Event handler for the receipt of a non-response command from the transport.

Parameters

command the received command object.

Implements `activemq::transport::TransportListener` (p. 3644).

6.23.3.35 `void activemq::core::ActiveMQConnection::oneway (Pointer< commands::Command > command) throw (activemq::exceptions::ActiveMQException)`

Sends a oneway message.

Parameters

command The message to send.

Exceptions

ConnectorException if not currently connected, or if the operation fails for any reason.

6.23.3.36 `virtual void activemq::core::ActiveMQConnection::onException (const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters

ex The exception.

Implements `activemq::transport::TransportListener` (p. 3645).

6.23.3.37 `virtual void activemq::core::ActiveMQConnection::removeDispatcher (const Pointer< commands::ConsumerId > & consumer) throw (cms::CMSException) [virtual]`

Removes the dispatcher for a consumer.

Parameters

consumer - The consumer for which to remove the dispatcher.

6.23.3.38 `virtual void activemq::core::ActiveMQConnection::removeProducer (const Pointer< commands::ProducerId > & producerId) throw (cms::CMSException) [virtual]`

Removes an active Producer to the Set of known producers.

Parameters

producerId - The ProducerId to remove from the the known set.

6.23.3.39 `virtual void activemq::core::ActiveMQConnection::removeSession (ActiveMQSession * session) throw (cms::CMSException) [virtual]`

Removes the session resources for the given session instance.

Parameters

session The session to be unregistered from this connection.

6.23.3.40 `void activemq::core::ActiveMQConnection::removeTransportListener (transport::TransportListener * transportListener)`

Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events.

The caller is responsible for freeing the listener in all cases.

Parameters

transportListener The pointer to the TransportListener to remove from the set of listeners.

6.23.3.41 `virtual void activemq::core::ActiveMQConnection::sendPullRequest (const commands::ConsumerInfo * consumer, long long timeout) throw (exceptions::ActiveMQException) [virtual]`

If supported sends a message pull request to the service provider asking for the delivery of a new message.

This is used in the case where the service provider has been configured with a zero prefetch or is only capable of delivering messages on a pull basis.

Parameters

consumer - the ConsumerInfo for the requesting Consumer.

timeout - the time that the client is willing to wait.

6.23.3.42 `void activemq::core::ActiveMQConnection::setAlwaysSyncSend (bool value)`

Sets if the Connection should always send things Synchronously.

Parameters

value true if sends should always be Synchronous.

6.23.3.43 `void activemq::core::ActiveMQConnection::setBrokerURL (const std::string & brokerURL)`

Sets the Broker URL that should be used when creating a new connection instance.

Parameters

brokerURL string

6.23.3.44 `virtual void activemq::core::ActiveMQConnection::setClientID (const std::string & clientID) [virtual]`

6.23.3.45 `void activemq::core::ActiveMQConnection::setCloseTimeout (unsigned int timeout)`

Sets the close timeout to use when sending the disconnect request.

Parameters

timeout - The time to wait for a close message.

6.23.3.46 `void activemq::core::ActiveMQConnection::setDefaultClientId (const std::string & clientId)`

Sets the Client Id.

Parameters

clientId - The new clientId value.

6.23.3.47 `void activemq::core::ActiveMQConnection::setDispatchAsync (bool value)`

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

For slow consumers set it to true so that dispatching will not block fast consumers. .

Parameters

value The value of the dispatch asynchronously option sent to the broker.

6.23.3.48 `virtual void activemq::core::ActiveMQConnection::setExceptionListener (cms::ExceptionListener * listener) [virtual]`

Sets the registered Exception Listener for this connection.

Parameters

listener pointer to and ExceptionListener

6.23.3.49 void activemq::core::ActiveMQConnection::setPassword (const std::string & *password*)

Sets the password that should be used when creating a new connection.

Parameters

password string

6.23.3.50 void activemq::core::ActiveMQConnection::setPrefetchPolicy (PrefetchPolicy * *policy*)

Sets the **PrefetchPolicy** (p. 2783) instance that this factory should use when it creates new Connection instances.

The **PrefetchPolicy** (p. 2783) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

policy The new **PrefetchPolicy** (p. 2783) that the ConnectionFactory should clone for Connections.

6.23.3.51 void activemq::core::ActiveMQConnection::setProducerWindowSize (unsigned int *windowSize*)

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

Parameters

windowSize - The size in bytes of the Producers memory window.

6.23.3.52 void activemq::core::ActiveMQConnection::setRedeliveryPolicy (RedeliveryPolicy * *policy*)

Sets the **RedeliveryPolicy** (p. 2972) instance that this factory should use when it creates new Connection instances.

The **RedeliveryPolicy** (p. 2972) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

policy The new **RedeliveryPolicy** (p. 2972) that the ConnectionFactory should clone for Connections.

6.23.3.53 void activemq::core::ActiveMQConnection::setSendTimeout (unsigned int *timeout*)

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

Parameters

timeout - The time to wait for a response.

6.23.3.54 `void activemq::core::ActiveMQConnection::setTransportInterruptionProcessingComplete ()`

Indicates that a Connection resource that is processing the transportInterrupted event has completed.

6.23.3.55 `void activemq::core::ActiveMQConnection::setUseAsyncSend (bool value)`

Sets the useAsyncSend option.

Parameters

value - true to activate, false to disable.

6.23.3.56 `void activemq::core::ActiveMQConnection::setUseCompression (bool value)`

Sets whether Message body compression is enabled.

Parameters

value Boolean indicating if Message body compression is enabled.

6.23.3.57 `void activemq::core::ActiveMQConnection::setUsername (const std::string & username)`

Sets the username that should be used when creating a new connection.

Parameters

username string

6.23.3.58 `virtual void activemq::core::ActiveMQConnection::start () throw (cms::CMSException) [virtual]`

Starts or (restarts) a connections delivery of incoming messages.

Exceptions

CMSException

6.23.3.59 `virtual void activemq::core::ActiveMQConnection::stop () throw (cms::CMSException) [virtual]`

Stop the flow of incoming messages.

Exceptions

CMSException

6.23.3.60 `void activemq::core::ActiveMQConnection::syncRequest (Pointer< commands::Command > command, unsigned int timeout = 0) throw (activemq::exceptions::ActiveMQException)`

Sends a synchronous request and returns the response from the broker.

Converts any error responses into an exception.

Parameters

command The request command.

timeout The time to wait for a response, default is zero or infinite.

Exceptions

ConnectorException thrown if an error response was received from the broker, or if any other error occurred.

6.23.3.61 `virtual void activemq::core::ActiveMQConnection::transportInterrupted () [virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements `activemq::transport::TransportListener` (p. 3645).

6.23.3.62 `virtual void activemq::core::ActiveMQConnection::transportResumed () [virtual]`

The transport has resumed after an interruption.

Implements `activemq::transport::TransportListener` (p. 3645).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnectionFactory.h`

6.24 activemq::core::ActiveMQConnectionFactory Class Reference

```
#include <src/main/activemq/core/ActiveMQConnectionFactory.h>
```

Inheritance diagram for `activemq::core::ActiveMQConnectionFactory`:

Public Member Functions

- **ActiveMQConnectionFactory** ()
- **ActiveMQConnectionFactory** (const std::string &url, const std::string &username="", const std::string &password="")
Constructor.
- virtual ~**ActiveMQConnectionFactory** ()
- virtual **cms::Connection * createConnection** () throw (cms::CMSException)
Creates a connection with the default user identity.
- virtual **cms::Connection * createConnection** (const std::string &username, const std::string &password) throw (cms::CMSException)
Creates a connection with the specified user identity.
- virtual **cms::Connection * createConnection** (const std::string &username, const std::string &password, const std::string &clientId) throw (cms::CMSException)
Creates a connection with the specified user identity.
- void **setUsername** (const std::string &username)
Sets the username that should be used when creating a new connection.
- const std::string & **getUsername** () const
Gets the username that this factory will use when creating a new connection instance.
- void **setPassword** (const std::string &password)
Sets the password that should be used when creating a new connection.
- const std::string & **getPassword** () const
Gets the password that this factory will use when creating a new connection instance.
- std::string **getClientId** () const
Gets the Configured Client Id.
- void **setClientId** (const std::string &clientId)
Sets the Client Id.
- void **setBrokerURL** (const std::string &brokerURL)
Sets the Broker URL that should be used when creating a new connection instance.
- const std::string & **getBrokerURL** () const
Gets the Broker URL that this factory will use when creating a new connection instance.
- void **setExceptionListener** (cms::ExceptionListener *listener)
Set an CMS ExceptionListener that will be set on eat connection once it has been created.
- **cms::ExceptionListener * getExceptionListener** () const
Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.

- void **setPrefetchPolicy** (**PrefetchPolicy** *policy)
*Sets the **PrefetchPolicy** (p. 2783) instance that this factory should use when it creates new Connection instances.*
- **PrefetchPolicy** * **getPrefetchPolicy** () const
*Gets the pointer to the current **PrefetchPolicy** (p. 2783) that is in use by this ConnectionFactory.*
- void **setRedeliveryPolicy** (**RedeliveryPolicy** *policy)
*Sets the **RedeliveryPolicy** (p. 2972) instance that this factory should use when it creates new Connection instances.*
- **RedeliveryPolicy** * **getRedeliveryPolicy** () const
*Gets the pointer to the current **RedeliveryPolicy** (p. 2972) that is in use by this ConnectionFactory.*
- bool **isDispatchAsync** () const
- void **setDispatchAsync** (bool value)
Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.
- bool **isAlwaysSyncSend** () const
Gets if the Connection should always send things Synchronously.
- void **setAlwaysSyncSend** (bool value)
Sets if the Connection should always send things Synchronously.
- bool **isUseAsyncSend** () const
Gets if the useAsyncSend option is set.
- void **setUseAsyncSend** (bool value)
Sets the useAsyncSend option.
- bool **isUseCompression** () const
Gets if the Connection is configured for Message body compression.
- void **setUseCompression** (bool value)
Sets whether Message body compression is enabled.
- unsigned int **getSendTimeout** () const
Gets the assigned send timeout for this Connector.
- void **setSendTimeout** (unsigned int timeout)
Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.
- unsigned int **getCloseTimeout** () const
Gets the assigned close timeout for this Connector.
- void **setCloseTimeout** (unsigned int timeout)
Sets the close timeout to use when sending the disconnect request.

- unsigned int **getProducerWindowSize** () const
Gets the configured producer window size for Producers that are created from this connector.
- void **setProducerWindowSize** (unsigned int windowSize)
Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

Static Public Member Functions

- static **cms::Connection * createConnection** (const std::string &url, const std::string &username, const std::string &password, const std::string &clientId="") throw (cms::CMSException)
Creates a connection with the specified user identity.

Static Public Attributes

- static const std::string **DEFAULT_URI**

6.24.1 Constructor & Destructor Documentation

- 6.24.1.1** **activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory**
()
- 6.24.1.2** **activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory**
(const std::string & *url*, const std::string & *username* = "", const std::string & *password* = "")

Constructor.

Parameters

url the URL of the Broker we are connecting to.
username to authenticate with, defaults to ""
password to authenticate with, defaults to ""

- 6.24.1.3** **virtual**
activemq::core::ActiveMQConnectionFactory::~~ActiveMQConnectionFactory
 () [virtual]

6.24.2 Member Function Documentation

- 6.24.2.1** **virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection** () throw (cms::CMSException) [virtual]

Creates a connection with the default user identity.

The connection is created in stopped mode. No messages will be delivered until the `Connection.start` method is explicitly called.

Returns

a Connection Pointer

Exceptions

CMSEException

Implements `cms::ConnectionFactory` (p. 1230).

6.24.2.2 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & username, const std::string & password) throw (cms::CMSEException) [virtual]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the `Connection.start` method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless `createConnection` will continue to use the default values that were set in the Constructor.

Parameters

username to authenticate with

password to authenticate with

Returns

a Connection Pointer

Exceptions

CMSEException

Implements `cms::ConnectionFactory` (p. 1229).

6.24.2.3 `static cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & url, const std::string & username, const std::string & password, const std::string & clientId = "") throw (cms::CMSEException) [static]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the `Connection.start` method is explicitly called.

Parameters

url the URL of the Broker we are connecting to.

username to authenticate with

password to authenticate with
clientId to assign to connection, defaults to ""

Exceptions

CMSEException.

6.24.2.4 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & username, const std::string & password, const std::string & clientId) throw (cms::CMSEException) [virtual]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the `Connection.start` method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless `createConnection` will continue to use the default values that were set in the Constructor.

Parameters

username to authenticate with
password to authenticate with
clientId to assign to connection if "" then a random client Id is created for this connection.

Returns

a Connection Pointer

Exceptions

CMSEException

Implements `cms::ConnectionFactory` (p. 1230).

6.24.2.5 `const std::string& activemq::core::ActiveMQConnectionFactory::getBrokerURL () const`

Gets the Broker URL that this factory will use when creating a new connection instance.

Returns

brokerURL string

6.24.2.6 `std::string activemq::core::ActiveMQConnectionFactory::getClientId () const`

Gets the Configured Client Id.

Returns

the clientId.

6.24.2.7 `unsigned int activemq::core::ActiveMQConnectionFactory::getCloseTimeout () const`

Gets the assigned close timeout for this Connector.

Returns

the close timeout configured in the connection uri

6.24.2.8 `cms::ExceptionListener* activemq::core::ActiveMQConnectionFactory::getExceptionListener () const`

Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.

Returns

a pointer to a CMS ExceptionListener instance or NULL if not set.

6.24.2.9 `const std::string& activemq::core::ActiveMQConnectionFactory::getPassword () const`

Gets the password that this factory will use when creating a new connection instance.

Returns

password string, "" for default credentials

6.24.2.10 `PrefetchPolicy* activemq::core::ActiveMQConnectionFactory::getPrefetchPolicy () const`

Gets the pointer to the current **PrefetchPolicy** (p. 2783) that is in use by this ConnectionFactory.

Returns

a pointer to this objects **PrefetchPolicy** (p. 2783).

6.24.2.11 `unsigned int activemq::core::ActiveMQConnectionFactory::getProducerWindowSize () const`

Gets the configured producer window size for Producers that are created from this connector. This only applies if there is no send timeout and the producer is able to send asynchronously.

Returns

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

6.24.2.12 `RedeliveryPolicy* activemq::core::ActiveMQConnectionFactory::getRedeliveryPolicy () const`

Gets the pointer to the current **RedeliveryPolicy** (p. 2972) that is in use by this **ConnectionFactory**.

Returns

a pointer to this objects **RedeliveryPolicy** (p. 2972).

6.24.2.13 `unsigned int activemq::core::ActiveMQConnectionFactory::getSendTimeout () const`

Gets the assigned send timeout for this **Connector**.

Returns

the send timeout configured in the connection uri

6.24.2.14 `const std::string& activemq::core::ActiveMQConnectionFactory::getUsername () const`

Gets the username that this factory will use when creating a new connection instance.

Returns

username string, "" for default credentials

6.24.2.15 `bool activemq::core::ActiveMQConnectionFactory::isAlwaysSyncSend () const`

Gets if the **Connection** should always send things **Synchronously**.

Returns

true if sends should always be **Synchronous**.

6.24.2.16 `bool activemq::core::ActiveMQConnectionFactory::isDispatchAsync () const`

Returns

The value of the dispatch asynchronously option sent to the broker.

6.24.2.17 `bool activemq::core::ActiveMQConnectionFactory::isUseAsyncSend () const`

Gets if the useAsyncSend option is set.

Returns

true if on false if not.

6.24.2.18 `bool activemq::core::ActiveMQConnectionFactory::isUseCompression () const`

Gets if the Connection is configured for Message body compression.

Returns

if the Message body will be Compressed or not.

6.24.2.19 `void activemq::core::ActiveMQConnectionFactory::setAlwaysSyncSend (bool value)`

Sets if the Connection should always send things Synchronously.

Parameters

value true if sends should always be Synchronous.

6.24.2.20 `void activemq::core::ActiveMQConnectionFactory::setBrokerURL (const std::string & brokerURL)`

Sets the Broker URL that should be used when creating a new connection instance.

Parameters

brokerURL string

6.24.2.21 `void activemq::core::ActiveMQConnectionFactory::setClientId (const std::string & clientId)`

Sets the Client Id.

Parameters

clientId - The new clientId value.

6.24.2.22 `void activemq::core::ActiveMQConnectionFactory::setCloseTimeout (unsigned int timeout)`

Sets the close timeout to use when sending the disconnect request.

Parameters

timeout - The time to wait for a close message.

6.24.2.23 void activemq::core::ActiveMQConnectionFactory::setDispatchAsync (bool *value*)

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

For slow consumers set it to true so that dispatching will not block fast consumers. .

Parameters

value The value of the dispatch asynchronously option sent to the broker.

6.24.2.24 void activemq::core::ActiveMQConnectionFactory::setExceptionListener (cms::ExceptionListener * *listener*)

Set an CMS ExceptionListener that will be set on eat connection once it has been created.

The factory does not take ownership of this pointer, the client must ensure that its lifetime is scoped to the connection that it is applied to.

Parameters

listener The listener to set on the connection or NULL for no listener.

6.24.2.25 void activemq::core::ActiveMQConnectionFactory::setPassword (const std::string & *password*)

Sets the password that should be used when creating a new connection.

Parameters

password string

6.24.2.26 void activemq::core::ActiveMQConnectionFactory::setPrefetchPolicy (PrefetchPolicy * *policy*)

Sets the **PrefetchPolicy** (p.2783) instance that this factory should use when it creates new Connection instances.

The **PrefetchPolicy** (p.2783) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

policy The new **PrefetchPolicy** (p.2783) that the ConnectionFactory should clone for Connections.

6.24.2.27 void activemq::core::ActiveMQConnectionFactory::setProducerWindowSize (unsigned int *windowSize*)

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

Parameters

windowSize - The size in bytes of the Producers memory window.

6.24.2.28 void activemq::core::ActiveMQConnectionFactory::setRedeliveryPolicy (RedeliveryPolicy * *policy*)

Sets the **RedeliveryPolicy** (p. 2972) instance that this factory should use when it creates new Connection instances.

The **RedeliveryPolicy** (p. 2972) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

policy The new **RedeliveryPolicy** (p. 2972) that the ConnectionFactory should clone for Connections.

6.24.2.29 void activemq::core::ActiveMQConnectionFactory::setSendTimeout (unsigned int *timeout*)

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

Parameters

timeout - The time to wait for a response.

6.24.2.30 void activemq::core::ActiveMQConnectionFactory::setUseAsyncSend (bool *value*)

Sets the useAsyncSend option.

Parameters

value - true to activate, false to disable.

6.24.2.31 void activemq::core::ActiveMQConnectionFactory::setUseCompression (bool *value*)

Sets whether Message body compression is enabled.

Parameters

value Boolean indicating if Message body compression is enabled.

6.24.2.32 void activemq::core::ActiveMQConnectionFactory::setUsername (const std::string & *username*)

Sets the username that should be used when creating a new connection.

Parameters

username string

6.24.3 Field Documentation

- 6.24.3.1 `const std::string`
`activemq::core::ActiveMQConnectionFactory::DEFAULT_`
`URI [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnectionFactory.h`

6.25 `activemq::core::ActiveMQConnectionMetaData` Class Reference

This class houses all the various settings and information that is used by an instance of an `ActiveMQConnection` (p. 233) class.

```
#include <src/main/activemq/core/ActiveMQConnectionMetaData.h>
```

Inheritance diagram for `activemq::core::ActiveMQConnectionMetaData`:

Public Member Functions

- `ActiveMQConnectionMetaData ()`
- `virtual ~ActiveMQConnectionMetaData ()`
- `virtual std::string getCMSVersion () const throw (cms::CMSEException)`
Gets the CMS API version.
- `virtual int getCMSMajorVersion () const throw (cms::CMSEException)`
Gets the CMS major version number.
- `virtual int getCMSMinorVersion () const throw (cms::CMSEException)`
Gets the CMS minor version number.
- `virtual std::string getCMSProviderName () const throw (cms::CMSEException)`
Gets the CMS provider name.
- `virtual std::string getProviderVersion () const throw (cms::CMSEException)`
Gets the CMS provider version.
- `virtual int getProviderMajorVersion () const throw (cms::CMSEException)`
Gets the CMS provider major version number.
- `virtual int getProviderMinorVersion () const throw (cms::CMSEException)`
Gets the CMS provider minor version number.
- `virtual std::vector< std::string > getCMSXPropertyNames () const throw (cms::CMSEException)`
Gets an Vector of the CMSX property names.

6.25.1 Detailed Description

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 233) class.

Since

3.0

6.25.2 Constructor & Destructor Documentation

6.25.2.1 `activemq::core::ActiveMQConnectionMetaData::ActiveMQConnectionMetaData ()`

6.25.2.2 `virtual
activemq::core::ActiveMQConnectionMetaData::~~ActiveMQConnectionMetaData
() [virtual]`

6.25.3 Member Function Documentation

6.25.3.1 `virtual int ac-
tivemq::core::ActiveMQConnectionMetaData::getCMSMajorVersion ()
const throw (cms::CMSException) [virtual]`

Gets the CMS major version number.

Returns

the CMS API major version number

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 1288).

6.25.3.2 `virtual int ac-
tivemq::core::ActiveMQConnectionMetaData::getCMSMinorVersion ()
const throw (cms::CMSException) [virtual]`

Gets the CMS minor version number.

Returns

the CMS API minor version number

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 1289).

6.25.3.3 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSProviderName () const throw (cms::CMSException) [virtual]`

Gets the CMS provider name.

Returns

the CMS provider name

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1289).

6.25.3.4 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSVersion () const throw (cms::CMSException) [virtual]`

Gets the CMS API version.

Returns

the CMS API Version in String form.

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1289).

6.25.3.5 `virtual std::vector<std::string> activemq::core::ActiveMQConnectionMetaData::getCMSXPropertyNames () const throw (cms::CMSException) [virtual]`

Gets an Vector of the CMSX property names.

Returns

an Vector of CMSX property names

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1290).

6.25.3.6 `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMajorVersion () const throw (cms::CMSException) [virtual]`

Gets the CMS provider major version number.

Returns

the CMS provider major version number

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1290).

6.25.3.7 `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMinorVersion () const throw (cms::CMSException) [virtual]`

Gets the CMS provider minor version number.

Returns

the CMS provider minor version number

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1290).

6.25.3.8 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getProviderVersion () const throw (cms::CMSException) [virtual]`

Gets the CMS provider version.

Returns

the CMS provider version

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1290).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnectionMetaData.h`

6.26 activemq::core::ActiveMQConstants Class Reference

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

Data Structures

- class `StaticInitializer`

Public Types

- enum `TransactionState` {
`TRANSACTION_STATE_BEGIN` = 0, `TRANSACTION_STATE_PREPARE` = 1, `TRANSACTION_STATE_COMMITONEPHASE` = 2, `TRANSACTION_STATE_COMMITTWOPHASE` = 3,
`TRANSACTION_STATE_ROLLBACK` = 4, `TRANSACTION_STATE_RECOVER` = 5, `TRANSACTION_STATE_FORGET` = 6, `TRANSACTION_STATE_END` = 7 }
 - enum `DestinationActions` { `DESTINATION_ADD_OPERATION` = 0, `DESTINATION_REMOVE_OPERATION` = 1 }
 - enum `AckType` {
`ACK_TYPE_DELIVERED` = 0, `ACK_TYPE_POISON` = 1, `ACK_TYPE_CONSUMED` = 2, `ACK_TYPE_REDELIVERED` = 3,
`ACK_TYPE_INDIVIDUAL` = 4 }
 - enum `DestinationOption` {
`CONSUMER_PREFETCHSIZE`, `CONSUMER_MAXPENDINGMSGLIMIT`, `CONSUMER_NOLOCAL`, `CONSUMER_DISPATCHASYNC`,
`CONSUMER_RETROACTIVE`, `CONSUMER_SELECTOR`, `CONSUMER_EXCLUSIVE`, `CONSUMER_PRIORITY`,
`NUM_OPTIONS` }
- These values represent the options that can be appended to an Destination name, i.e.*
- enum `URIParam` {
`CONNECTION_SENDTIMEOUT`, `CONNECTION_PRODUCERWINDOWSIZE`, `CONNECTION_CLOSETIMEOUT`,
`CONNECTION_ALWAYSSENDCONNECTION_USEASYNCSEND`, `CONNECTION_USECOMPRESSION`,
`CONNECTION_DISPATCHASYNC`, `PARAM_USERNAME`,
`PARAM_PASSWORD`, `PARAM_CLIENTID`, `NUM_PARAMS` }
- These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.*

Static Public Member Functions

- static const std::string & **toString** (const **DestinationOption** option)
- static **DestinationOption** **toDestinationOption** (const std::string &option)
- static const std::string & **toString** (const **URIParam** option)
- static **URIParam** **toURIOption** (const std::string &option)

6.26.1 Detailed Description

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

6.26.2 Member Enumeration Documentation

6.26.2.1 enum activemq::core::ActiveMQConstants::AckType

Enumerator:

```
ACK_TYPE_DELIVERED
ACK_TYPE_POISON
ACK_TYPE_CONSUMED
ACK_TYPE_REDELIVERED
ACK_TYPE_INDIVIDUAL
```

6.26.2.2 enum activemq::core::ActiveMQConstants::DestinationActions

Enumerator:

```
DESTINATION_ADD_OPERATION
DESTINATION_REMOVE_OPERATION
```

6.26.2.3 enum activemq::core::ActiveMQConstants::DestinationOption

These values represent the options that can be appended to an Destination name, i.e.
/topic/foo?consumer.exclusive=true

Enumerator:

```
CONSUMER_PREFETCHSIZE
CUNSUMER_MAXPENDINGMSGLIMIT
CONSUMER_NOLOCAL
CONSUMER_DISPATCHASYNC
CONSUMER_RETROACTIVE
CONSUMER_SELECTOR
CONSUMER_EXCLUSIVE
CONSUMER_PRIORITY
NUM_OPTIONS
```

6.26.2.4 enum activemq::core::ActiveMQConstants::TransactionState

Enumerator:

```
TRANSACTION_STATE_BEGIN
TRANSACTION_STATE_PREPARE
TRANSACTION_STATE_COMMITONEPHASE
TRANSACTION_STATE_COMMITTWOPHASE
TRANSACTION_STATE_ROLLBACK
TRANSACTION_STATE_RECOVER
TRANSACTION_STATE_FORGET
TRANSACTION_STATE_END
```

6.26.2.5 enum activemq::core::ActiveMQConstants::URIParam

These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.

Enumerator:

```
CONNECTION_SENDDTIMEOUT
CONNECTION_PRODUCERWINDOWSIZE
CONNECTION_CLOSETIMEOUT
CONNECTION_ALWAYSASYNCSEND
CONNECTION_USEASYNCSEND
CONNECTION_USECOMPRESSION
CONNECTION_DISPATCHASYNC
PARAM_USERNAME
PARAM_PASSWORD
PARAM_CLIENTID
NUM_PARAMS
```

6.26.3 Member Function Documentation

- 6.26.3.1 static DestinationOption activemq::core::ActiveMQConstants::toDestinationOption (const std::string & *option*) [inline, static]
- 6.26.3.2 static const std::string& activemq::core::ActiveMQConstants::toString (const DestinationOption *option*) [inline, static]
- 6.26.3.3 static const std::string& activemq::core::ActiveMQConstants::toString (const URIParam *option*) [inline, static]
- 6.26.3.4 static URIParam activemq::core::ActiveMQConstants::toURIOption (const std::string & *option*) [inline, static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQConstants.h

6.27 activemq::core::ActiveMQConsumer Class Reference

```
#include <src/main/activemq/core/ActiveMQConsumer.h>
```

Inheritance diagram for activemq::core::ActiveMQConsumer:

Public Member Functions

- **ActiveMQConsumer** (**ActiveMQSession** *session, const **Pointer**< **commands::ConsumerId** > &id, const **Pointer**< **commands::ActiveMQDestination** > &destination, const std::string &name, const std::string &selector, int prefetch, int maxPendingMessageCount, bool noLocal, bool browser, bool dispatchAsync, **cms::MessageListener** *listener)

Constructor.

- virtual ~**ActiveMQConsumer** ()
- virtual void **start** ()

Starts the Consumer if not already started and not closed.

- virtual void **stop** ()

Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again.

- virtual void **close** () throw (cms::CMSEException)

Closes the Consumer.

- virtual **cms::Message** * **receive** () throw (cms::CMSEException)

Synchronously Receive a Message.

- virtual **cms::Message** * **receive** (int millisecs) throw (cms::CMSEException)

Synchronously Receive a Message, time out after defined interval.

- virtual **cms::Message** * **receiveNoWait** () throw (cms::CMSEException)

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

- virtual void **setMessageListener** (**cms::MessageListener** *listener) throw (cms::CMSEException)

Sets the MessageListener that this class will send notifs on.

- virtual **cms::MessageListener** * **getMessageListener** () const throw (cms::CMSEException)

Gets the MessageListener that this class will send events to.

- virtual std::string **getMessageSelector** () const throw (cms::CMSEException)

Gets this message consumer's message selector expression.

- virtual void **acknowledge** (const **Pointer**< **commands::MessageDispatch** > &dispatch) throw (cms::CMSEException)

Method called to acknowledge the message passed, called from a message when the mode is client ack.

- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)
Called asynchronously by the session to dispatch a message.
- void **acknowledge** () throw (cms::CMSException)
Method called to acknowledge all messages that have been received so far.
- void **commit** () throw (exceptions::ActiveMQException)
Called to Commit the current set of messages in this Transaction.
- void **rollback** () throw (exceptions::ActiveMQException)
Called to Roll back the current set of messages in this Transaction.
- void **doClose** () throw (exceptions::ActiveMQException)
Performs the actual close operation on this consumer.
- const **Pointer**< **commands::ConsumerInfo** > & **getConsumerInfo** () const
Get the Consumer information for this consumer.
- const **Pointer**< **commands::ConsumerId** > & **getConsumerId** () const
Get the Consumer Id for this consumer.
- bool **isClosed** () const
- bool **isSynchronizationRegistered** () const
*Has this Consumer Transaction **Synchronization** (p. 3477) been added to the transaction.*
- void **setSynchronizationRegistered** (bool value)
*Sets the **Synchronization** (p. 3477) Registered state of this consumer.*
- bool **iterate** ()
Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.
- void **deliverAcks** () throw (exceptions::ActiveMQException)
Forces this consumer to send all pending acks to the broker.
- void **clearMessagesInProgress** ()
Called on a Failover to clear any pending messages.
- void **inProgressClearRequired** ()
Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.
- long long **getLastDeliveredSequenceId** () const
Gets the currently set Last Delivered Sequence Id.
- void **setLastDeliveredSequenceId** (long long value)
Sets the value of the Last Delivered Sequence Id.

- `int getMessageAvailableCount () const`
- `void setRedeliveryPolicy (RedeliveryPolicy *policy)`
*Sets the **RedeliveryPolicy** (p. 2972) this Consumer should use when a rollback is performed on a transacted Consumer.*
- `RedeliveryPolicy * getRedeliveryPolicy () const`
Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.

Protected Member Functions

- `Pointer< MessageDispatch > dequeue (long long timeout) throw (cms::CMSException)`
Used by synchronous receive methods to wait for messages to come in.
- `void beforeMessageIsConsumed (const Pointer< commands::MessageDispatch > &dispatch)`
Pre-consume processing.
- `void afterMessageIsConsumed (const Pointer< commands::MessageDispatch > &dispatch, bool messageExpired)`
Post-consume processing.

6.27.1 Constructor & Destructor Documentation

- 6.27.1.1** `activemq::core::ActiveMQConsumer::ActiveMQConsumer (ActiveMQSession * session, const Pointer< commands::ConsumerId > & id, const Pointer< commands::ActiveMQDestination > & destination, const std::string & name, const std::string & selector, int prefetch, int maxPendingMessageCount, bool noLocal, bool browser, bool dispatchAsync, cms::MessageListener * listener)`

Constructor.

- 6.27.1.2** `virtual activemq::core::ActiveMQConsumer::~~ActiveMQConsumer ()`
`[virtual]`

6.27.2 Member Function Documentation

- 6.27.2.1** `virtual void activemq::core::ActiveMQConsumer::acknowledge (const Pointer< commands::MessageDispatch > & dispatch) throw (cms::CMSException)` `[virtual]`

Method called to acknowledge the message passed, called from a message when the mode is client ack.

Parameters

message the Message to Acknowledge

Exceptions*CMSEException*

6.27.2.2 `void activemq::core::ActiveMQConsumer::acknowledge () throw (cms::CMSEException)`

Method called to acknowledge all messages that have been received so far.

Exceptions*CMSEException*

6.27.2.3 `void activemq::core::ActiveMQConsumer::afterMessageIsConsumed (const Pointer< commands::MessageDispatch > & dispatch, bool messageExpired) [protected]`

Post-consume processing.

Parameters

dispatch - the consumed message

messageExpired - flag indicating if the message has expired.

6.27.2.4 `void activemq::core::ActiveMQConsumer::beforeMessageIsConsumed (const Pointer< commands::MessageDispatch > & dispatch) [protected]`

Pre-consume processing.

Parameters

dispatch - the message being consumed.

6.27.2.5 `void activemq::core::ActiveMQConsumer::clearMessagesInProgress ()`

Called on a Failover to clear any pending messages.

6.27.2.6 `virtual void activemq::core::ActiveMQConsumer::close () throw (cms::CMSEException) [virtual]`

Closes the Consumer.

This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

Exceptions*CMSEException*

6.27.2.7 void activemq::core::ActiveMQConsumer::commit () throw (exceptions::ActiveMQException)

Called to Commit the current set of messages in this Transaction.

Exceptions

ActiveMQException

6.27.2.8 void activemq::core::ActiveMQConsumer::deliverAcks () throw (exceptions::ActiveMQException)

Forces this consumer to send all pending acks to the broker.

6.27.2.9 Pointer<MessageDispatch> activemq::core::ActiveMQConsumer::dequeue (long long *timeout*) throw (cms::CMSException) [protected]

Used by synchronous receive methods to wait for messages to come in.

Parameters

timeout - The maximum number of milliseconds to wait before returning. If -1, it will block until a messages is received or this consumer is closed. If 0, will not block at all. If > 0, will wait at a maximum the specified number of milliseconds before returning.

Returns

the message, if received within the allotted time. Otherwise NULL.

Exceptions

InvalidStateException if this consumer is closed upon entering this method.

6.27.2.10 virtual void activemq::core::ActiveMQConsumer::dispatch (const Pointer< MessageDispatch > & *message*) [virtual]

Called asynchronously by the session to dispatch a message.

Parameters

message dispatch object pointer

Implements **activemq::core::Dispatcher** (p. 1672).

6.27.2.11 void activemq::core::ActiveMQConsumer::doClose () throw (exceptions::ActiveMQException)

Performs the actual close operation on this consumer.

Exceptions

ActiveMQException

6.27.2.12 `const Pointer<commands::ConsumerId>& activemq::core::ActiveMQConsumer::getConsumerId () const [inline]`

Get the Consumer Id for this consumer.

Returns

Reference to a Consumer Id Object

6.27.2.13 `const Pointer<commands::ConsumerInfo>& activemq::core::ActiveMQConsumer::getConsumerInfo () const [inline]`

Get the Consumer information for this consumer.

Returns

Reference to a Consumer Info Object

6.27.2.14 `long long activemq::core::ActiveMQConsumer::getLastDeliveredSequenceId () const [inline]`

Gets the currently set Last Delivered Sequence Id.

Returns

long long containing the sequence id of the last delivered Message.

6.27.2.15 `int activemq::core::ActiveMQConsumer::getMessageAvailableCount () const`

Returns

the number of Message's this consumer is waiting to Dispatch.

6.27.2.16 `virtual cms::MessageListener* activemq::core::ActiveMQConsumer::getMessageListener () const throw (cms::CMSException) [inline, virtual]`

Gets the MessageListener that this class will send events to.

Returns

the currently registered MessageListener interface pointer.

6.27.2.17 `virtual std::string activemq::core::ActiveMQConsumer::getMessageSelector () const throw (cms::CMSException) [virtual]`

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

cms::CMSException (p. 1074)

6.27.2.18 `RedeliveryPolicy* activemq::core::ActiveMQConsumer::getRedeliveryPolicy () const [inline]`

Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.

Returns

a Pointer to a **RedeliveryPolicy** (p. 2972) that is in use by this Consumer.

6.27.2.19 `void activemq::core::ActiveMQConsumer::inProgressClearRequired ()`

Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.

6.27.2.20 `bool activemq::core::ActiveMQConsumer::isClosed () const [inline]`

Returns

if this Consumer has been closed.

6.27.2.21 `bool activemq::core::ActiveMQConsumer::isSynchronizationRegistered () const [inline]`

Has this Consumer Transaction **Synchronization** (p. 3477) been added to the transaction.

Returns

true if the synchronization has been added.

6.27.2.22 `bool activemq::core::ActiveMQConsumer::iterate ()`

Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.

6.27.2.23 `virtual cms::Message* activemq::core::ActiveMQConsumer::receive ()
throw (cms::CMSException) [virtual]`

Synchronously Receive a Message.

Returns

new message

Exceptions

CMSException

6.27.2.24 `virtual cms::Message* activemq::core::ActiveMQConsumer::receive (int
millisecs) throw (cms::CMSException) [virtual]`

Synchronously Receive a Message, time out after defined interval.

Returns null if nothing read.

Parameters

millisecs the time in milliseconds to wait before returning

Returns

new message or null on timeout

Exceptions

CMSException

6.27.2.25 `virtual cms::Message* ac-
tivemq::core::ActiveMQConsumer::receiveNoWait ()
throw (cms::CMSException) [virtual]`

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns

new message

Exceptions

CMSException

6.27.2.26 `void activemq::core::ActiveMQConsumer::rollback () throw (exceptions::ActiveMQException)`

Called to Roll back the current set of messages in this Transaction.

Exceptions

ActiveMQException

6.27.2.27 `void activemq::core::ActiveMQConsumer::setLastDeliveredSequenceId (long long value) [inline]`

Sets the value of the Last Delivered Sequence Id.

Parameters

value The new value to assign to the Last Delivered Sequence Id property.

6.27.2.28 `virtual void activemq::core::ActiveMQConsumer::setMessageListener (cms::MessageListener * listener) throw (cms::CMSException) [virtual]`

Sets the MessageListener that this class will send notifs on.

Parameters

listener MessageListener interface pointer

6.27.2.29 `void activemq::core::ActiveMQConsumer::setRedeliveryPolicy (RedeliveryPolicy * policy) [inline]`

Sets the **RedeliveryPolicy** (p. 2972) this Consumer should use when a rollback is performed on a transacted Consumer.

The Consumer takes ownership of the passed pointer. The Consumer's redelivery policy can never be null, a call to this method with a NULL pointer is ignored.

Parameters

policy Pointer to a Redelivery Policy object that his Consumer will use.

6.27.2.30 `void activemq::core::ActiveMQConsumer::setSynchronizationRegistered (bool value) [inline]`

Sets the **Synchronization** (p. 3477) Registered state of this consumer.

Parameters

value - true if registered false otherwise.

6.27.2.31 `virtual void activemq::core::ActiveMQConsumer::start () [virtual]`

Starts the Consumer if not already started and not closed.

A consumer will no deliver messages until started.

6.27.2.32 virtual void activemq::core::ActiveMQConsumer::stop () [virtual]

Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again.

A Closed Consumer is also a stopped consumer.

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConsumer.h**

6.28 activemq::library::ActiveMQCPP Class Reference

```
#include <src/main/activemq/library/ActiveMQCPP.h>
```

Public Member Functions

- virtual **~ActiveMQCPP** ()

Static Public Member Functions

- static void **initializeLibrary** ()

Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf library.

- static void **initializeLibrary** (int argc, char **argv)

Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library.

- static void **shutdownLibrary** ()

Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.

Protected Member Functions

- **ActiveMQCPP** ()
- **ActiveMQCPP** (const **ActiveMQCPP** &)
- **ActiveMQCPP** & **operator=** (const **ActiveMQCPP** &)

6.28.1 Constructor & Destructor Documentation

6.28.1.1 `activemq::library::ActiveMQCPP::ActiveMQCPP ()` [inline, protected]

6.28.1.2 `activemq::library::ActiveMQCPP::ActiveMQCPP (const ActiveMQCPP &)` [protected]

6.28.1.3 `virtual activemq::library::ActiveMQCPP::~~ActiveMQCPP ()` [inline, virtual]

6.28.2 Member Function Documentation

6.28.2.1 `static void activemq::library::ActiveMQCPP::initializeLibrary ()` [static]

Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf library.

Exceptions

runtime_error if an error occurs while initializing this library.

6.28.2.2 `static void activemq::library::ActiveMQCPP::initializeLibrary (int argc, char ** argv)` [static]

Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library.

This method takes the args passed to the main method of process for use is setting system properties and configuring the ActiveMQ-CPP Library.

Parameters

argc - the count of arguments passed to this Process.

argv - the array of string arguments passed to this process.

Exceptions

runtime_error if an error occurs while initializing this library.

6.28.2.3 `ActiveMQCPP& activemq::library::ActiveMQCPP::operator= (const ActiveMQCPP &)` [protected]

6.28.2.4 `static void activemq::library::ActiveMQCPP::shutdownLibrary ()` [static]

Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.

All the user created ActiveMQ-CPP objects and Decaf Library objects should have been destroyed by the time this method is called.

The documentation for this class was generated from the following file:

- `src/main/activemq/library/ActiveMQCPP.h`

6.29 `activemq::commands::ActiveMQDestination` Class Reference

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

Inheritance diagram for `activemq::commands::ActiveMQDestination`:

Data Structures

- struct `DestinationFilter`

Public Member Functions

- `ActiveMQDestination ()`
- `ActiveMQDestination (const std::string &physicalName)`
- virtual `~ActiveMQDestination ()`
- virtual `ActiveMQDestination * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void `copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual bool `equals (const DataStructure *value) const`
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual unsigned char `getDataStructureType () const`
*Get the **DataStructure** (p. 1553) Type as defined in *CommandTypes.h*.*
- virtual std::string `toString () const`
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual const std::string & `getPhysicalName () const`
Fetch this destination's physical name.
- virtual std::string & `getPhysicalName ()`
- virtual void `setPhysicalName (const std::string &physicalName)`
Set this destination's physical name.
- virtual bool `isAdvisory () const`
- virtual void `setAdvisory (bool advisory)`
- virtual bool `isConsumerAdvisory () const`
- virtual bool `isProducerAdvisory () const`
- virtual bool `isConnectionAdvisory () const`

- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool **exclusive**)
- virtual bool **isOrdered** () const
- virtual void **setOrdered** (bool **ordered**)
- virtual std::string **getOrderedTarget** () const
- virtual void **setOrderedTarget** (const std::string &**orderedTarget**)
- virtual **cms::Destination::DestinationType** **getDestinationType** () const =0
Returns the Type of Destination that this object represents.
- virtual bool **isTemporary** () const
Returns true if a temporary Destination.
- virtual bool **isTopic** () const
Returns true if a Topic Destination.
- virtual bool **isQueue** () const
Returns true if a Queue Destination.
- virtual bool **isComposite** () const
Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.
- virtual bool **isWildcard** () const
- const **activemq::util::ActiveMQProperties** & **getOptions** () const
- virtual const **cms::Destination** * **getCMSDestination** () const

Static Public Member Functions

- static std::string **createTemporaryName** (const std::string &clientId)
Create a temporary name from the clientId.
- static std::string **getClientId** (const **ActiveMQDestination** *destination)
From a temporary destination find the clientId of the Connection that created it.
- static **Pointer< ActiveMQDestination >** **createDestination** (int type, const std::string &name)
Creates a Destination given the String Name to use and a Type.

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQDESTINATION** = 0

Protected Attributes

- bool **exclusive**
- bool **ordered**
- bool **advisory**
- std::string **orderedTarget**
- std::string **physicalName**
- **util::ActiveMQProperties** **options**

Static Protected Attributes

- static const std::string **ADVISORY_PREFIX**
prefix for Advisory message destinations
- static const std::string **CONSUMER_ADVISORY_PREFIX**
prefix for consumer advisory destinations
- static const std::string **PRODUCER_ADVISORY_PREFIX**
prefix for producer advisory destinations
- static const std::string **CONNECTION_ADVISORY_PREFIX**
prefix for connection advisory destinations
- static const std::string **DEFAULT_ORDERED_TARGET**
The default target for ordered destinations.
- static const std::string **TEMP_PREFIX**
- static const std::string **TEMP_POSTFIX**
- static const std::string **COMPOSITE_SEPARATOR**
- static const std::string **QUEUE_QUALIFIED_PREFIX**
- static const std::string **TOPIC_QUALIFIED_PREFIX**
- static const std::string **TEMP_QUEUE_QUALIFIED_PREFIX**
- static const std::string **TEMP_TOPIC_QUALIFIED_PREFIX**

6.29.1 Constructor & Destructor Documentation

6.29.1.1 `activemq::commands::ActiveMQDestination::ActiveMQDestination ()`

6.29.1.2 `activemq::commands::ActiveMQDestination::ActiveMQDestination (const std::string & physicalName)`

6.29.1.3 `virtual
activemq::commands::ActiveMQDestination::~~ActiveMQDestination ()
[inline, virtual]`

6.29.2 Member Function Documentation

6.29.2.1 `virtual ActiveMQDestination* ac-
tivemq::commands::ActiveMQDestination::cloneDataStructure () const
[inline, virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.29.2.2 `virtual void activemq::commands::ActiveMQDestination::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1555).

Referenced by **activemq::commands::ActiveMQTempDestination::copyDataStructure()**.

6.29.2.3 `static Pointer<ActiveMQDestination> activemq::commands::ActiveMQDestination::createDestination (int type, const std::string & name) [static]`

Creates a Destination given the String Name to use and a Type.

Parameters

type - The Type of Destination to Create

name - The Name to use in the creation of the Destination

Returns

Pointer to a new **ActiveMQDestination** (p. 279) instance.

6.29.2.4 `static std::string activemq::commands::ActiveMQDestination::createTemporaryName (const std::string & clientId) [inline, static]`

Create a temporary name from the clientId.

Parameters

clientId

Returns

6.29.2.5 `virtual bool activemq::commands::ActiveMQDestination::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1556).

Referenced by **activemq::commands::ActiveMQTopic::equals()**, and **activemq::commands::ActiveMQTempDestination::equals()**.

6.29.2.6 `static std::string activemq::commands::ActiveMQDestination::getClientId (const ActiveMQDestination * destination) [static]`

From a temporary destination find the clientId of the Connection that created it.

Parameters

destination

Returns

the clientId or null if not a temporary destination

6.29.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQDestination::getCMSDestination () const [inline, virtual]`

Returns

the **cms::Destination** (p. 1610) interface pointer that the objects that derive from this class implement.

6.29.2.8 `virtual unsigned char activemq::commands::ActiveMQDestination::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1553) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1557).

6.29.2.9 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQDestination::getDestinationType () const [pure virtual]`

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

6.29.2.10 `const activemq::util::ActiveMQProperties& activemq::commands::ActiveMQDestination::getOptions () const`
[inline]

Returns

a reference (const) to the options properties for this Destination.

6.29.2.11 `virtual std::string activemq::commands::ActiveMQDestination::getOrderedTarget () const`
[inline, virtual]

Returns

Returns the orderedTarget.

6.29.2.12 `virtual const std::string& activemq::commands::ActiveMQDestination::getPhysicalName () const`
[inline, virtual]

Fetch this destination's physical name.

Returns

const string containing the name

6.29.2.13 `virtual std::string& activemq::commands::ActiveMQDestination::getPhysicalName ()`
[inline, virtual]

6.29.2.14 `virtual bool activemq::commands::ActiveMQDestination::isAdvisory () const` [inline, virtual]

Returns

Returns the advisory.

6.29.2.15 `virtual bool activemq::commands::ActiveMQDestination::isComposite () const` [inline, virtual]

Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.

Returns

true if this destination represents a collection of child destinations.

6.29.2.16 `virtual bool activemq::commands::ActiveMQDestination::isConnectionAdvisory ()`
`const [inline, virtual]`

Returns

true if this is a destination for Connection advisories

6.29.2.17 `virtual bool activemq::commands::ActiveMQDestination::isConsumerAdvisory ()`
`const [inline, virtual]`

Returns

true if this is a destination for Consumer advisories

6.29.2.18 `virtual bool activemq::commands::ActiveMQDestination::isExclusive ()`
`const [inline, virtual]`

Returns

Returns the exclusive.

6.29.2.19 `virtual bool activemq::commands::ActiveMQDestination::isOrdered ()`
`const [inline, virtual]`

Returns

Returns the ordered.

6.29.2.20 `virtual bool activemq::commands::ActiveMQDestination::isProducerAdvisory ()`
`const [inline, virtual]`

Returns

true if this is a destination for Producer advisories

6.29.2.21 `virtual bool activemq::commands::ActiveMQDestination::isQueue ()`
`const [inline, virtual]`

Returns true if a Queue Destination.

Returns

true/false

6.29.2.22 `virtual bool activemq::commands::ActiveMQDestination::isTemporary () const [inline, virtual]`

Returns true if a temporary Destination.

Returns

true/false

References cms::Destination::TEMPORARY_TOPIC.

6.29.2.23 `virtual bool activemq::commands::ActiveMQDestination::isTopic () const [inline, virtual]`

Returns true if a Topic Destination.

Returns

true/false

References cms::Destination::TOPIC.

6.29.2.24 `virtual bool activemq::commands::ActiveMQDestination::isWildcard () const [inline, virtual]`

Returns

true if the destination matches multiple possible destinations

6.29.2.25 `virtual void activemq::commands::ActiveMQDestination::setAdvisory (bool advisory) [inline, virtual]`

Parameters

advisory The advisory to set.

6.29.2.26 `virtual void activemq::commands::ActiveMQDestination::setExclusive (bool exclusive) [inline, virtual]`

Parameters

exclusive The exclusive to set.

6.29.2.27 `virtual void activemq::commands::ActiveMQDestination::setOrdered (bool ordered) [inline, virtual]`

Parameters

ordered The ordered to set.

6.29.2.28 `virtual void activemq::commands::ActiveMQDestination::setOrderedTarget (const std::string & orderedTarget)` [inline, virtual]

Parameters

orderedTarget The orderedTarget to set.

6.29.2.29 `virtual void activemq::commands::ActiveMQDestination::setPhysicalName (const std::string & physicalName)` [virtual]

Set this destination's physical name.

Returns

const string containing the name

6.29.2.30 `virtual std::string activemq::commands::ActiveMQDestination::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.767).

6.29.3 Field Documentation

6.29.3.1 `bool activemq::commands::ActiveMQDestination::advisory` [protected]

6.29.3.2 `const std::string activemq::commands::ActiveMQDestination::ADVISORY_PREFIX` [static, protected]

prefix for Advisory message destinations

6.29.3.3 `const std::string activemq::commands::ActiveMQDestination::COMPOSITE_SEPARATOR` [static, protected]

6.29.3.4 `const std::string activemq::commands::ActiveMQDestination::CONNECTION_ADVISORY_PREFIX` [static, protected]

prefix for connection advisory destinations

6.29.3.5 `const std::string`
`activemq::commands::ActiveMQDestination::CONSUMER_`
`ADVISORY_PREFIX` `[static, protected]`

prefix for consumer advisory destinations

6.29.3.6 `const std::string`
`activemq::commands::ActiveMQDestination::DEFAULT_`
`ORDERED_TARGET` `[static, protected]`

The default target for ordered destinations.

6.29.3.7 `bool activemq::commands::ActiveMQDestination::exclusive` `[protected]`

6.29.3.8 `const unsigned char activemq::commands::ActiveMQDestination::ID_`
`ACTIVEMQDESTINATION = 0` `[static]`

6.29.3.9 `util::ActiveMQProperties ac-`
`tivemq::commands::ActiveMQDestination::options`
`[protected]`

6.29.3.10 `bool activemq::commands::ActiveMQDestination::ordered` `[protected]`

6.29.3.11 `std::string activemq::commands::ActiveMQDestination::orderedTarget`
`[protected]`

6.29.3.12 `std::string activemq::commands::ActiveMQDestination::physicalName`
`[protected]`

6.29.3.13 `const std::string`
`activemq::commands::ActiveMQDestination::PRODUCER_`
`ADVISORY_PREFIX` `[static, protected]`

prefix for producer advisory destinations

- 6.29.3.14 `const std::string activemq::commands::ActiveMQDestination::QUEUE_ - QUALIFIED_PREFIX` [static, protected]
- 6.29.3.15 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - POSTFIX` [static, protected]
- 6.29.3.16 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - PREFIX` [static, protected]
- 6.29.3.17 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - QUEUE_QUALIFIED_PREFIX` [static, protected]
- 6.29.3.18 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - TOPIC_QUALIFIED_PREFIX` [static, protected]
- 6.29.3.19 `const std::string activemq::commands::ActiveMQDestination::TOPIC_ - QUALIFIED_PREFIX` [static, protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQDestination.h`

6.30 `activemq::wireformat::openwire::marshal::v3::ActiveMQDestination` Class Reference

Marshaling code for Open Wire Format for `ActiveMQDestinationMarshaller` (p. 289).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller`:

Public Member Functions

- `ActiveMQDestinationMarshaller ()`
- `virtual ~ActiveMQDestinationMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.30.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 289).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.30.2 Constructor & Destructor Documentation

6.30.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller () [inline]

6.30.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller () [inline, virtual]

6.30.3 Member Function Documentation

6.30.3.1 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1518).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 443), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**

(p. 528), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 554), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 582), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 638).

6.30.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 443), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 528), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 555), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 583), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 639).

6.30.3.3 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 443), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 529), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 555), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 583), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 639).

6.30.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal2
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataOutputStream * *dataOut*,**
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 444), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 529), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 555), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 583), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 639).

6.30.3.5 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightUnmarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataInputStream * *dataIn*,**
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 444), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 530), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 556), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 584), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 640).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h`

6.31 **activemq::wireformat::openwire::marshal::v1::ActiveMQDestination** Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 293).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.31.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 293).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.31.2 Constructor & Destructor Documentation

6.31.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller () [inline]

6.31.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller () [inline, virtual]

6.31.3 Member Function Documentation

6.31.3.1 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1518).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 447), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 531), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 558), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 590), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 646).

```

6.31.3.2 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::looseUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 447), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 532), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 559), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 591), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 647).

```

6.31.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshall
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 447), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 532), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 559), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 591), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 647).

6.31.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 448), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 533), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 559), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 591), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 647).

6.31.3.5 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 448), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 533), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 560), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller`

(p. 592), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 648).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h`

6.32 `activemq::wireformat::openwire::marshal::v4::ActiveMQDestination` Class Reference

Marshaling code for Open Wire Format for `ActiveMQDestinationMarshaller` (p. 297).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller`:

Public Member Functions

- `ActiveMQDestinationMarshaller ()`
- `virtual ~ActiveMQDestinationMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write the booleans that this object uses to a BooleanStream.

- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write a object instance to data output stream.

- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.32.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 297).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.32.2 Constructor & Destructor Documentation

6.32.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller () [inline]`

6.32.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller () [inline, virtual]`

6.32.3 Member Function Documentation

6.32.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 451), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 535), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 562), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 586), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 642).

6.32.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 451), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 536), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 563), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 587), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 643).

```
6.32.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 451), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 536), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 563), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 587), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 643).

```
6.32.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 452), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 537), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 563), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 587), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 643).

6.32.3.5 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 452), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 537), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 564), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 588), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 644).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h`

6.33 activemq::wireformat::openwire::marshal::v5::ActiveMQDestination Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 301).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller`:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.33.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 301).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.33.2 Constructor & Destructor Documentation

6.33.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller ()` [inline]

6.33.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller ()` [inline, virtual]

6.33.3 Member Function Documentation

6.33.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 455), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 539), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 566), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 594), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 650).

6.33.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 455), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 539), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 567), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 595), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 651).

```
6.33.3.3 virtual int ac-
          tivemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal1
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
          ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 455), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 540), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 567), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 595), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 651).

```
6.33.3.4 virtual void ac-
          tivemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal2
          ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 456), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 540), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 567), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 595), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 651).

6.33.3.5 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 456), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 541), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 568), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 596), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 652).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h`

6.34 `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQDestinationMarshaller` (p. 305).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.34.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 305).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.34.2 Constructor & Destructor Documentation

6.34.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller () [inline]`

6.34.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller () [inline, virtual]`

6.34.3 Member Function Documentation

6.34.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 459), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 542), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 570), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 598), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 658).

6.34.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 459), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 543), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 571), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 599), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 659).

```
6.34.3.3 virtual int ac-
          tivemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal1
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
          ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 459), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 543), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 571), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 599), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 659).

```
6.34.3.4 virtual void ac-
          tivemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal2
          ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink

bs - `BooleanStream` stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 460), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 544), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 571), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 599), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 659).

6.34.3.5 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - `BinaryReader` that provides that data.

bs - `BooleanStream` stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 460), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 544), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 572), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 600), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 660).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h`

6.35 `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQDestinationMarshaller` (p. 309).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller`:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.35.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 309).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.35.2 Constructor & Destructor Documentation

6.35.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller () [inline]`

6.35.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller () [inline, virtual]`

6.35.3 Member Function Documentation

6.35.3.1 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller` (p. 463), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 546), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 574), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 602), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller` (p. 654).

6.35.3.2 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller` (p. 463), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 547), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 575), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 603), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller` (p. 655).

```
6.35.3.3 virtual int ac-
          tivemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightMarshal1
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
          ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller` (p. 463), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 547), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 575), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 603), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller` (p. 655).

```
6.35.3.4 virtual void ac-
          tivemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightMarshal2
          ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller` (p. 464), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 548), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 575), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 603), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller` (p. 655).

```
6.35.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller` (p. 464), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 548), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 576), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 604), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller` (p. 656).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h`

6.36 activemq::exceptions::ActiveMQException Class Reference

```
#include <src/main/activemq/exceptions/ActiveMQException.h>
```

Inheritance diagram for `activemq::exceptions::ActiveMQException`:

Public Member Functions

- **ActiveMQException** () throw ()
Default Constructor.
- **ActiveMQException** (const **ActiveMQException** &ex) throw ()
Copy Constructor.
- **ActiveMQException** (const **decaf::lang::Exception** &ex) throw ()
Copy Constructor.
- **ActiveMQException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual ~**ActiveMQException** () throw ()
- virtual **ActiveMQException** * **clone** () const
Clones this exception.
- virtual **cms::CMSEException** **convertToCMSEException** () const
Converts this exception to a new CMSEException.

6.36.1 Constructor & Destructor Documentation

6.36.1.1 **activemq::exceptions::ActiveMQException::ActiveMQException** () throw ()

Default Constructor.

6.36.1.2 **activemq::exceptions::ActiveMQException::ActiveMQException** (const **ActiveMQException** & *ex*) throw ()

Copy Constructor.

Parameters

ex The Exception whose internal data is copied into this instance.

6.36.1.3 **activemq::exceptions::ActiveMQException::ActiveMQException** (const **decaf::lang::Exception** & *ex*) throw ()

Copy Constructor.

Parameters

ex The Exception whose internal data is copied into this instance.

6.36.1.4 `activemq::exceptions::ActiveMQException::ActiveMQException (const char * file, const int lineNumber, const char * msg, ...) throw ()`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

Parameters

file The file name where exception occurs.

lineNumber The line number where the exception occurred.

msg The message to report.

... The list of primitives that are formatted into the message.

6.36.1.5 `virtual activemq::exceptions::ActiveMQException::~~ActiveMQException () throw () [virtual]`

6.36.2 Member Function Documentation

6.36.2.1 `virtual ActiveMQException* activemq::exceptions::ActiveMQException::clone () const [virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

Copy of this Exception object

Reimplemented from `decaf::lang::Exception` (p. 1715).

Reimplemented in `activemq::exceptions::BrokerException` (p. 798).

6.36.2.2 `virtual cms::CMSException activemq::exceptions::ActiveMQException::convertToCMSException () const [virtual]`

Converts this exception to a new CMSException.

Returns

a CMSException with the data from this exception

The documentation for this class was generated from the following file:

- `src/main/activemq/exceptions/ActiveMQException.h`

6.37 activemq::commands::ActiveMQMapMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQMapMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQMapMessage:

Public Member Functions

- **ActiveMQMapMessage** ()
- virtual **~ActiveMQMapMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual bool **isMarshalAware** () const
Determine if this object is aware of marshaling and should have its before and after marshaling methods called.
- virtual **ActiveMQMapMessage** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat) throw (decaf::io::IOException)
Perform any processing needed before an marshal.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw (cms::CMSEException)
Clears out the body of the message.
- virtual **cms::MapMessage** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual std::vector< std::string > **getMapNames** () const throw (cms::CMSEException)
Returns an Enumeration of all the names in the MapMessage object.
- virtual bool **itemExists** (const std::string &name) const throw (cms::CMSEException)
Indicates whether an item exists in this MapMessage object.

- virtual bool **getBoolean** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Boolean value of the Specified name.
- virtual void **setBoolean** (const std::string &name, bool value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a boolean value with the specified name into the Map.
- virtual unsigned char **getByte** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Byte value of the Specified name.
- virtual void **setByte** (const std::string &name, unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Byte value with the specified name into the Map.
- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Bytes value of the Specified name.
- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Bytes value with the specified name into the Map.
- virtual char **getChar** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Char value of the Specified name.
- virtual void **setChar** (const std::string &name, char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Char value with the specified name into the Map.
- virtual double **getDouble** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Double value of the Specified name.
- virtual void **setDouble** (const std::string &name, double value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Double value with the specified name into the Map.
- virtual float **getFloat** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Float value of the Specified name.
- virtual void **setFloat** (const std::string &name, float value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Float value with the specified name into the Map.
- virtual int **getInt** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Int value of the Specified name.

- virtual void **setInt** (const std::string &name, int value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Int value with the specified name into the Map.
- virtual long long **getLong** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Long value of the Specified name.
- virtual void **setLong** (const std::string &name, long long value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Long value with the specified name into the Map.
- virtual short **getShort** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Short value of the Specified name.
- virtual void **setShort** (const std::string &name, short value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Short value with the specified name into the Map.
- virtual std::string **getString** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the String value of the Specified name.
- virtual void **setString** (const std::string &name, const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a String value with the specified name into the Map.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQMAPMESSAGE** = 25

Protected Member Functions

- **util::PrimitiveMap & getMap** () throw (decaf::lang::exceptions::NullPointerException)
Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.
- const **util::PrimitiveMap & getMap** () const throw (decaf::lang::exceptions::NullPointerException)
- virtual void **checkMapIsUnmarshalled** () const throw (decaf::lang::exceptions::NullPointerException)
Performs the unmarshal on the Map if needed, otherwise just returns.

6.37.1 Constructor & Destructor Documentation

6.37.1.1 `activemq::commands::ActiveMQMapMessage::ActiveMQMapMessage ()`

6.37.1.2 `virtual
activemq::commands::ActiveMQMapMessage::~~ActiveMQMapMessage ()` [virtual]

6.37.2 Member Function Documentation

6.37.2.1 `virtual void activemq::commands::ActiveMQMapMessage::beforeMarshal (wireformat::WireFormat * wireFormat) throw (decaf::io::IOException)` [virtual]

Perform any processing needed before an marshal.

Parameters

wireFormat - the OpenWireFormat object in use.

Implements `activemq::wireformat::MarshalAware` (p. 2329).

6.37.2.2 `virtual void activemq::commands::ActiveMQMapMessage::checkMapIsUnmarshalled () const throw (decaf::lang::exceptions::NullPointerException)` [protected, virtual]

Performs the unmarshal on the Map if needed, otherwise just returns.

6.37.2.3 `virtual void activemq::commands::ActiveMQMapMessage::clearBody () throw (cms::CMSException)` [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 383).

6.37.2.4 `virtual cms::MapMessage* activemq::commands::ActiveMQMapMessage::clone () const` [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.37.2.5 `virtual ActiveMQMapMessage* activemq::commands::ActiveMQMapMessage::cloneDataStructure ()`
`const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2363).

6.37.2.6 `virtual void activemq::commands::ActiveMQMapMessage::copyDataStructure (const DataStructure * src)`
`[virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2363).

6.37.2.7 `virtual bool activemq::commands::ActiveMQMapMessage::equals (const DataStructure * value)`
`const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 383).

6.37.2.8 `virtual bool activemq::commands::ActiveMQMapMessage::getBoolean (const std::string & name)`
`const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Boolean value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.37.2.9 virtual unsigned char activemq::commands::ActiveMQMapMessage::getByte (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSException) [virtual]

Returns the Byte value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.37.2.10 virtual std::vector<unsigned char> activemq::commands::ActiveMQMapMessage::getBytes (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSException) [virtual]

Returns the Bytes value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.37.2.11 virtual char activemq::commands::ActiveMQMapMessage::getChar (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSException) [virtual]

Returns the Char value of the Specified name.

Parameters

name name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.37.2.12 `virtual unsigned char activemq::commands::ActiveMQMapMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Reimplemented from **activemq::commands::Message** (p. 2365).

6.37.2.13 `virtual double activemq::commands::ActiveMQMapMessage::getDouble (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Double value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.37.2.14 `virtual float activemq::commands::ActiveMQMapMessage::getFloat (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Float value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.37.2.15 `virtual int activemq::commands::ActiveMQMapMessage::getInt (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Int value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.37.2.16 `virtual long long activemq::commands::ActiveMQMapMessage::getLong (const std::string & name) const throw (cms::MessageFormatException, cms::CMSEException) [virtual]`

Returns the Long value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.37.2.17 `util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap () throw (decaf::lang::exceptions::NullPointerException) [protected]`

Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.

Returns

reference to a PrimitiveMap.

6.37.2.18 `const util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap () const throw (decaf::lang::exceptions::NullPointerException) [protected]`

6.37.2.19 `virtual std::vector< std::string > activemq::commands::ActiveMQMapMessage::getMapNames () const throw (cms::CMSEException) [virtual]`

Returns an Enumeration of all the names in the MapMessage object.

Returns

STL Vector of String values, each of which is the name of an item in the MapMessage

Exceptions

CMSEException - if the operation fails due to an internal error.

6.37.2.20 `virtual short activemq::commands::ActiveMQMapMessage::getShort (const std::string & name) const throw (cms::MessageFormatException, cms::CMSEException) [virtual]`

Returns the Short value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.37.2.21 `virtual std::string activemq::commands::ActiveMQMapMessage::getString (const std::string & name) const throw (cms::MessageFormatException, cms::CMSEException) [virtual]`

Returns the String value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.37.2.22 `virtual bool activemq::commands::ActiveMQMapMessage::isMarshalAware () const [inline, virtual]`

Determine if this object is aware of marshaling and should have its before and after marshaling methods called.

Defaults to false.

Returns

true if aware of marshaling

Reimplemented from `activemq::commands::Message` (p. 2368).

6.37.2.23 `virtual bool activemq::commands::ActiveMQMapMessage::itemExists (const std::string & name) const throw (cms::CMSEException) [virtual]`

Indicates whether an item exists in this MapMessage object.

Parameters

name String name of the Object in question

Returns

boolean value indicating if the name is in the map

Exceptions

CMSEException - if the operation fails due to an internal error.

6.37.2.24 `virtual void activemq::commands::ActiveMQMapMessage::setBoolean (const std::string & name, bool value) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a boolean value with the specified name into the Map.

Parameters

name the name of the boolean

value the boolean value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the Message (p. 2358) is in Read-only Mode.

6.37.2.25 `virtual void activemq::commands::ActiveMQMapMessage::setByte (const std::string & name, unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Byte value with the specified name into the Map.

Parameters

name the name of the Byte

value the Byte value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the Message (p. 2358) is in Read-only Mode.

6.37.2.26 `virtual void activemq::commands::ActiveMQMapMessage::setBytes (const std::string & name, const std::vector< unsigned char > & value) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Bytes value with the specified name into the Map.

Parameters

name The name of the Bytes

value The Bytes value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the **Message** (p. 2358) is in Read-only Mode.

6.37.2.27 `virtual void activemq::commands::ActiveMQMapMessage::setChar
(const std::string & name, char value) throw (
cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Char value with the specified name into the Map.

Parameters

name the name of the Char

value the Char value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the **Message** (p. 2358) is in Read-only Mode.

6.37.2.28 `virtual void activemq::commands::ActiveMQMapMessage::setDouble
(const std::string & name, double value) throw (
cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Double value with the specified name into the Map.

Parameters

name The name of the Double

value The Double value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the **Message** (p. 2358) is in Read-only Mode.

6.37.2.29 `virtual void activemq::commands::ActiveMQMapMessage::setFloat
(const std::string & name, float value) throw (
cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Float value with the specified name into the Map.

Parameters

name The name of the Float

value The Float value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the *Message* (p. 2358) is in Read-only Mode.

6.37.2.30 virtual void activemq::commands::ActiveMQMapMessage::setInt
(const std::string & *name*, int *value*) throw (
cms::MessageNotWriteableException, cms::CMSEException) [virtual]

Sets a Int value with the specified name into the Map.

Parameters

name The name of the Int

value The Int value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the *Message* (p. 2358) is in Read-only Mode.

6.37.2.31 virtual void activemq::commands::ActiveMQMapMessage::setLong
(const std::string & *name*, long long *value*) throw (
cms::MessageNotWriteableException, cms::CMSEException) [virtual]

Sets a Long value with the specified name into the Map.

Parameters

name The name of the Long

value The Long value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the *Message* (p. 2358) is in Read-only Mode.

6.37.2.32 virtual void activemq::commands::ActiveMQMapMessage::setShort
(const std::string & *name*, short *value*) throw (
cms::MessageNotWriteableException, cms::CMSEException) [virtual]

Sets a Short value with the specified name into the Map.

Parameters

name The name of the Short

value The Short value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the *Message* (p. 2358) is in Read-only Mode.

6.37.2.33 `virtual void activemq::commands::ActiveMQMapMessage::setString
(const std::string & name, const std::string & value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Sets a String value with the specified name into the Map.

Parameters

name The name of the String

value The String value to set in the Map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageNotWriteableException - if the `Message` (p. 2358) is in Read-only Mode.

6.37.2.34 `virtual std::string activemq::commands::ActiveMQMapMessage::toString
() const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2372).

6.37.3 Field Documentation

6.37.3.1 `const unsigned char activemq::commands::ActiveMQMapMessage::ID_-
ACTIVEMQMAPMESSAGE = 25 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMapMessage.h`

6.38 `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessage` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 328).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.38.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 328).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.38.2 Constructor & Destructor Documentation

6.38.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller () [inline]`

6.38.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller () [inline, virtual]`

6.38.3 Member Function Documentation

6.38.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.38.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.38.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

6.38.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

6.38.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2530).

6.38.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2531).

```
6.38.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2531).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h`

6.39 `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 332).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller`:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.39.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 332).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.39.2 Constructor & Destructor Documentation

6.39.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller () [inline]`

6.39.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller () [inline, virtual]`

6.39.3 Member Function Documentation

6.39.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.39.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.39.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

6.39.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

6.39.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2542).

6.39.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543).

```
6.39.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightUnmars
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataInputStream * dataIn,
    utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h`

6.40 `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 336).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.40.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 336).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.40.2 Constructor & Destructor Documentation

6.40.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller () [inline]`

6.40.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller () [inline, virtual]`

6.40.3 Member Function Documentation

6.40.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.40.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.40.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

6.40.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

6.40.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2538).

6.40.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539).

```
6.40.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightUnmars
        ( OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataInputStream * dataIn,
        utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h`

6.41 `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 340).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.41.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 340).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.41.2 Constructor & Destructor Documentation

6.41.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` [inline]

6.41.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` [inline, virtual]

6.41.3 Member Function Documentation

6.41.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.41.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.41.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2524).

6.41.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2525).

6.41.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

6.41.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

```
6.41.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2527).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h`

6.42 `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 344).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.42.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 344).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.42.2 Constructor & Destructor Documentation

6.42.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller () [inline]`

6.42.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller () [inline, virtual]`

6.42.3 Member Function Documentation

6.42.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.42.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.42.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

6.42.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

6.42.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2534).

6.42.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535).

```
6.42.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h`

6.43 `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 348).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.43.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 348).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.43.2 Constructor & Destructor Documentation

6.43.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` [inline]

6.43.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` [inline, virtual]

6.43.3 Member Function Documentation

6.43.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.43.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.43.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

6.43.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

6.43.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2546).

6.43.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2547).

```
6.43.3.7 virtual void ac-
      tivemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightUnmars
      ( OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataInputStream * dataIn,
      utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2547).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMessageMarshaller.h`

6.44 `activemq::commands::ActiveMQMessage` Class Reference

```
#include <src/main/activemq/commands/ActiveMQMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQMessage`:

Public Member Functions

- **ActiveMQMessage** ()
- virtual **~ActiveMQMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual **ActiveMQMessage** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQMESSAGE** = 23

6.44.1 Constructor & Destructor Documentation

- 6.44.1.1 **activemq::commands::ActiveMQMessage::ActiveMQMessage** ()
- 6.44.1.2 **virtual activemq::commands::ActiveMQMessage::~~ActiveMQMessage** () [inline, virtual]

6.44.2 Member Function Documentation

- 6.44.2.1 **virtual cms::Message* activemq::commands::ActiveMQMessage::clone** () const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.44.2.2 `virtual ActiveMQMessage* activemq::commands::ActiveMQMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2363).

6.44.2.3 `virtual void activemq::commands::ActiveMQMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2363).

6.44.2.4 `virtual bool activemq::commands::ActiveMQMessage::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 383).

6.44.2.5 `virtual unsigned char activemq::commands::ActiveMQMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1553) type copy.

Reimplemented from `activemq::commands::Message` (p. 2365).

6.44.2.6 `virtual std::string activemq::commands::ActiveMQMessage::toString ()`
`const [virtual]`

Returns a string containing the information for this **DataSet** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2372).

6.44.3 Field Documentation

6.44.3.1 `const unsigned char activemq::commands::ActiveMQMessage::ID_ -`
`ACTIVEMQMESSAGE = 23 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMessage.h`

6.45 `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 355).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- `virtual ~ActiveMQMessageMarshaller` ()
- `virtual commands::DataSet * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataSet *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataSet *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.45.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 355). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.45.2 Constructor & Destructor Documentation

- 6.45.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller** () [inline]

- 6.45.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller** () [inline, virtual]

6.45.3 Member Function Documentation

- 6.45.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

- 6.45.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.45.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

6.45.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

6.45.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2530).

```
6.45.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2531).

```
6.45.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2531).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h`

6.46 `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMa` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMessageMarshaller` (p. 359).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller`:

Public Member Functions

- `ActiveMQMessageMarshaller ()`
- `virtual ~ActiveMQMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.46.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 359). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.46.2 Constructor & Destructor Documentation

6.46.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller () [inline]`

6.46.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller () [inline, virtual]`

6.46.3 Member Function Documentation

6.46.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.46.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.46.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

6.46.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

6.46.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2542).

```
6.46.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543).

```
6.46.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h`

6.47 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQMessageMarshaller` (p. 363).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`:

Public Member Functions

- `ActiveMQMessageMarshaller ()`
- `virtual ~ActiveMQMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.47.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 363). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.47.2 Constructor & Destructor Documentation

6.47.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller () [inline]`

6.47.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::~ActiveMQMessageMarshaller () [inline, virtual]`

6.47.3 Member Function Documentation

6.47.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.47.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.47.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

6.47.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

6.47.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2538).

```
6.47.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539).

```
6.47.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h`

6.48 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMa Class Reference

Marshaling code for Open Wire Format for `ActiveMQMessageMarshaller` (p. 367).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`:

Public Member Functions

- `ActiveMQMessageMarshaller ()`
- `virtual ~ActiveMQMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.48.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 367). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.48.2 Constructor & Destructor Documentation

6.48.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller () [inline]`

6.48.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::~ActiveMQMessageMarshaller () [inline, virtual]`

6.48.3 Member Function Documentation

6.48.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.48.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.48.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2524).

6.48.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2525).

6.48.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

```
6.48.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

```
6.48.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2527).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h`

6.49 `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMa` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMessageMarshaller` (p. 371).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`:

Public Member Functions

- `ActiveMQMessageMarshaller ()`
- `virtual ~ActiveMQMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.49.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 371). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.49.2 Constructor & Destructor Documentation

6.49.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller () [inline]`

6.49.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::~ActiveMQMessageMarshaller () [inline, virtual]`

6.49.3 Member Function Documentation

6.49.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.49.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.49.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

6.49.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

6.49.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2534).

```
6.49.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535).

```
6.49.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h`

6.50 activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQMessageMarshaller` (p. 375).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`:

Public Member Functions

- `ActiveMQMessageMarshaller ()`
- `virtual ~ActiveMQMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.50.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 375). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.50.2 Constructor & Destructor Documentation

6.50.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller () [inline]`

6.50.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::~ActiveMQMessageMarshaller () [inline, virtual]`

6.50.3 Member Function Documentation

6.50.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.50.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.50.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

6.50.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

6.50.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2546).

```
6.50.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2547).

```
6.50.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2547).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h

6.51 activemq::commands::ActiveMQMessageTemplate< T > Class Template Reference

```
#include <src/main/activemq/commands/ActiveMQMessageTemplate.h>
```

Inheritance diagram for activemq::commands::ActiveMQMessageTemplate< T >:

Public Member Functions

- **ActiveMQMessageTemplate** ()
- virtual **~ActiveMQMessageTemplate** ()
- virtual void **acknowledge** () const throw (cms::IllegalStateException, cms::CMSException)
Acknowledges all consumed messages of the session of this consumed message.
- virtual void **onSend** ()
*Resets the **Message** (p. 2358) to a Read-Only state.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw (cms::CMSException)
Clears out the body of the message.
- virtual void **clearProperties** () throw (cms::CMSException)
Clears the message properties.
- virtual std::vector< std::string > **getPropertyNames** () const throw (cms::CMSException)
Retrieves the property names.
- virtual bool **propertyExists** (const std::string &name) const throw (cms::CMSException)
Indicates whether or not a given property exists.
- virtual bool **getBooleanProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSException)

Gets a boolean property.

- virtual unsigned char **getByteProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a byte property.

- virtual double **getDoubleProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a double property.

- virtual float **getFloatProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a float property.

- virtual int **getIntProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a int property.

- virtual long long **getLongProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a long property.

- virtual short **getShortProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a short property.

- virtual std::string **getStringProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a string property.

- virtual void **setBooleanProperty** (const std::string &name, bool value) throw (cms::MessageNot WriteableException, cms::CMSEException)

Sets a boolean property.

- virtual void **setByteProperty** (const std::string &name, unsigned char value) throw (cms::MessageNot WriteableException, cms::CMSEException)

Sets a byte property.

- virtual void **setDoubleProperty** (const std::string &name, double value) throw (cms::MessageNot WriteableException, cms::CMSEException)

Sets a double property.

- virtual void **setFloatProperty** (const std::string &name, float value) throw (cms::MessageNot WriteableException, cms::CMSEException)

Sets a float property.

- virtual void **setIntProperty** (const std::string &name, int value) throw (cms::MessageNot WriteableException, cms::CMSEException)

Sets a int property.

- virtual void **setLongProperty** (const std::string &name, long long value) throw (cms::MessageNotWriteableException, cms::CMSException)
Sets a long property.
- virtual void **setShortProperty** (const std::string &name, short value) throw (cms::MessageNotWriteableException, cms::CMSException)
Sets a short property.
- virtual void **setStringProperty** (const std::string &name, const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSException)
Sets a string property.
- virtual std::string **getCMSCorrelationID** () const throw (cms::CMSException)
Get the Correlation Id for this message.
- virtual void **setCMSCorrelationID** (const std::string &correlationId) throw (cms::CMSException)
Sets the Correlation Id used by this message.
- virtual int **getCMSDeliveryMode** () const throw (cms::CMSException)
Gets the DeliveryMode for this message.
- virtual void **setCMSDeliveryMode** (int mode) throw (cms::CMSException)
Sets the DeliveryMode for this message.
- virtual const cms::Destination * **getCMSDestination** () const throw (cms::CMSException)
*Gets the Destination for this **Message** (p. 2358), returns a.*
- virtual void **setCMSDestination** (const cms::Destination *destination) throw (cms::CMSException)
Sets the Destination for this message.
- virtual long long **getCMSExpiration** () const throw (cms::CMSException)
*Gets the Expiration Time for this **Message** (p. 2358).*
- virtual void **setCMSExpiration** (long long expireTime) throw (cms::CMSException)
Sets the Expiration Time for this message.
- virtual std::string **getCMSMessageID** () const throw (cms::CMSException)
*Gets the CMS **Message** (p. 2358) Id for this **Message** (p. 2358).*
- virtual void **setCMSMessageID** (const std::string &id AMQCPP_UNUSED) throw (cms::CMSException)
*Sets the CMS **Message** (p. 2358) Id for this message.*
- virtual int **getCMSPriority** () const throw (cms::CMSException)
*Gets the Priority Value for this **Message** (p. 2358).*
- virtual void **setCMSPriority** (int priority) throw (cms::CMSException)

Sets the Priority Value for this message.

- virtual bool **getCMSRedelivered** () const throw (cms::CMSEException)

*Gets the Redelivered Flag for this **Message** (p. 2358).*

- virtual void **setCMSRedelivered** (bool redelivered AMQCPP_UNUSED) throw (cms::CMSEException)

Sets the Redelivered Flag for this message.

- virtual const cms::Destination * **getCMSReplyTo** () const throw (cms::CMSEException)

*Gets the CMS Reply To Address for this **Message** (p. 2358).*

- virtual void **setCMSReplyTo** (const cms::Destination *destination) throw (cms::CMSEException)

Sets the CMS Reply To Address for this message.

- virtual long long **getCMSTimestamp** () const throw (cms::CMSEException)

*Gets the Time Stamp for this **Message** (p. 2358).*

- virtual void **setCMSTimestamp** (long long timeStamp) throw (cms::CMSEException)

Sets the Time Stamp for this message.

- virtual std::string **getCMSType** () const throw (cms::CMSEException)

*Gets the CMS **Message** (p. 2358) Type for this **Message** (p. 2358).*

- virtual void **setCMSType** (const std::string &type) throw (cms::CMSEException)

*Sets the CMS **Message** (p. 2358) Type for this message.*

Protected Member Functions

- void **failIfWriteOnlyBody** () const
- void **failIfReadOnlyBody** () const
- void **failIfReadOnlyProperties** () const

```
template<typename T> class activemq::commands::ActiveMQMessageTemplate< T
>
```

6.51.1 Constructor & Destructor Documentation

6.51.1.1 `template<typename T> activemq::commands::ActiveMQMessageTemplate< T >::ActiveMQMessageTemplate () [inline]`

6.51.1.2 `template<typename T> virtual activemq::commands::ActiveMQMessageTemplate< T >::~~ActiveMQMessageTemplate () [inline, virtual]`

6.51.2 Member Function Documentation

6.51.2.1 `template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::acknowledge () const throw (cms::IllegalStateException, cms::CMSException) [inline, virtual]`

Acknowledges all consumed messages of the session of this consumed message.

6.51.2.2 `template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::clearBody () throw (cms::CMSException) [inline, virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented in `activemq::commands::ActiveMQBytesMessage` (p. 198), `activemq::commands::ActiveMQMapMessage` (p. 319), `activemq::commands::ActiveMQStreamMessage` (p. 489), and `activemq::commands::ActiveMQTextMessage` (p. 606).

6.51.2.3 `template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::clearProperties () throw (cms::CMSException) [inline, virtual]`

Clears the message properties.

Does not clear the body or header values.

6.51.2.4 `template<typename T> virtual bool activemq::commands::ActiveMQMessageTemplate< T >::equals (const DataStructure * value) const [inline, virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Message` (p. 2363).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 168), `activemq::commands::ActiveMQBytesMessage` (p. 199), `activemq::commands::ActiveMQMapMessage` (p. 320), `activemq::commands::ActiveMQMessage` (p. 354), `activemq::commands::ActiveMQObjectMessage` (p. 398), `activemq::commands::ActiveMQStreamMessage` (p. 490), and `activemq::commands::ActiveMQTextMessage` (p. 607).

```
6.51.2.5  template<typename T> void
          activemq::commands::ActiveMQMessageTemplate< T
          >::failIfReadOnlyBody (    ) const [inline, protected]
```

```
6.51.2.6  template<typename T> void
          activemq::commands::ActiveMQMessageTemplate< T
          >::failIfReadOnlyProperties (    ) const [inline, protected]
```

```
6.51.2.7  template<typename T> void
          activemq::commands::ActiveMQMessageTemplate< T
          >::failIfWriteOnlyBody (    ) const [inline, protected]
```

```
6.51.2.8  template<typename T> virtual bool
          activemq::commands::ActiveMQMessageTemplate< T
          >::getBooleanProperty ( const std::string & name ) const throw (
          cms::MessageFormatException, cms::CMSException ) [inline, virtual]
```

Gets a boolean property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.51.2.9  template<typename T> virtual unsigned char
          activemq::commands::ActiveMQMessageTemplate< T >::getByteProperty
          ( const std::string & name ) const throw ( cms::MessageFormatException,
          cms::CMSException ) [inline, virtual]
```

Gets a byte property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

6.51.2.10 `template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSCorrelationID () const throw (cms::CMSEException)
[inline, virtual]`

Get the Correlation Id for this message.

Returns

string representation of the correlation Id

Exceptions

CMSEException

6.51.2.11 `template<typename T> virtual int
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSDeliveryMode () const throw (cms::CMSEException)
[inline, virtual]`

Gets the DeliveryMode for this message.

Returns

DeliveryMode enumerated value.

Exceptions

CMSEException

6.51.2.12 `template<typename T> virtual const cms::Destination*
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSDestination () const throw (cms::CMSEException)
[inline, virtual]`

Gets the Destination for this **Message** (p. 2358), returns a.

Returns

Destination object

Exceptions

CMSEException

6.51.2.13 `template<typename T> virtual long long
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSExpiration () const throw (cms::CMSException) [inline,
virtual]`

Gets the Expiration Time for this **Message** (p. 2358).

Returns

time value

Exceptions

CMSException

6.51.2.14 `template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSMessageID () const throw (cms::CMSException)
[inline, virtual]`

Gets the CMS **Message** (p. 2358) Id for this **Message** (p. 2358).

Returns

time value

Exceptions

CMSException

6.51.2.15 `template<typename T> virtual int
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSPriority () const throw (cms::CMSException) [inline,
virtual]`

Gets the Priority Value for this **Message** (p. 2358).

Returns

priority value

Exceptions

CMSException

6.51.2.16 `template<typename T> virtual bool
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSRedelivered () const throw (cms::CMSException)
[inline, virtual]`

Gets the Redelivered Flag for this **Message** (p. 2358).

Returns

redelivered value

Exceptions

CMSException

6.51.2.17 `template<typename T> virtual const cms::Destination*
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSReplyTo () const throw (cms::CMSException) [inline,
virtual]`

Gets the CMS Reply To Address for this **Message** (p. 2358).

Returns

Reply To Value

Exceptions

CMSException

6.51.2.18 `template<typename T> virtual long long
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSTimestamp () const throw (cms::CMSException)
[inline, virtual]`

Gets the Time Stamp for this **Message** (p. 2358).

Returns

time stamp value

Exceptions

CMSException

6.51.2.19 `template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSType () const throw (cms::CMSException) [inline,
virtual]`

Gets the CMS **Message** (p. 2358) Type for this **Message** (p. 2358).

Returns

type value

Exceptions

CMSException

```
6.51.2.20  template<typename T> virtual double
           activemq::commands::ActiveMQMessageTemplate< T
           >::getDoubleProperty ( const std::string & name ) const throw (
           cms::MessageFormatException, cms::CMSException ) [inline, virtual]
```

Gets a double property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.51.2.21  template<typename T> virtual float
           activemq::commands::ActiveMQMessageTemplate< T
           >::getFloatProperty ( const std::string & name ) const throw (
           cms::MessageFormatException, cms::CMSException ) [inline, virtual]
```

Gets a float property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.51.2.22  template<typename T> virtual int
           activemq::commands::ActiveMQMessageTemplate< T
           >::getIntProperty ( const std::string & name ) const throw (
           cms::MessageFormatException, cms::CMSException ) [inline, virtual]
```

Gets a int property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.51.2.23  template<typename T> virtual long long
           activemq::commands::ActiveMQMessageTemplate< T
           >::getLongProperty ( const std::string & name ) const throw (
           cms::MessageFormatException, cms::CMSEException ) [inline, virtual]
```

Gets a long property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.51.2.24  template<typename T> virtual std::vector<std::string>
           activemq::commands::ActiveMQMessageTemplate< T
           >::getPropertyNames ( ) const throw ( cms::CMSEException ) [inline,
           virtual]
```

Retrieves the property names.

Returns

The complete set of property names currently in this message.

```
6.51.2.25  template<typename T> virtual short
           activemq::commands::ActiveMQMessageTemplate< T
           >::getShortProperty ( const std::string & name ) const throw (
           cms::MessageFormatException, cms::CMSEException ) [inline, virtual]
```

Gets a short property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

6.51.2.26 `template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getStringProperty (const std::string & name) const throw (
cms::MessageFormatException, cms::CMSException) [inline, virtual]`

Gets a string property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

6.51.2.27 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::onSend () [inline, virtual]`

Resets the **Message** (p. 2358) to a Read-Only state.

Reimplemented from **activemq::commands::Message** (p. 2369).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 200), and **activemq::commands::ActiveMQStreamMessage** (p. 490).

6.51.2.28 `template<typename T> virtual bool
activemq::commands::ActiveMQMessageTemplate< T
>::propertyExists (const std::string & name) const throw (
cms::CMSException) [inline, virtual]`

Indicates whether or not a given property exists.

Parameters

name The name of the property to look up.

Returns

True if the property exists in this message.

6.51.2.29 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setBooleanProperty (const std::string & name, bool value) throw (
cms::MessageNotWriteableException, cms::CMSException) [inline,
virtual]`

Sets a boolean property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

```
6.51.2.30  template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setByteProperty ( const std::string & name, unsigned char value )
throw ( cms::MessageNotWriteableException, cms::CMSEException )
[inline, virtual]
```

Sets a byte property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

```
6.51.2.31  template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSCorrelationID ( const std::string & correlationId ) throw (
cms::CMSEException ) [inline, virtual]
```

Sets the Correlation Id used by this message.

Parameters

correlationId - String representing the correlation id.

Exceptions

CMSEException

```
6.51.2.32  template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSDeliveryMode ( int mode ) throw ( cms::CMSEException )
[inline, virtual]
```

Sets the DeliveryMode for this message.

Parameters

mode - DeliveryMode enumerated value.

Exceptions

CMSException

```
6.51.2.33  template<typename T> virtual void
             activemq::commands::ActiveMQMessageTemplate< T
             >::setCMSDestination ( const cms::Destination * destination ) throw (
             cms::CMSException ) [inline, virtual]
```

Sets the Destination for this message.

Parameters

destination - Destination Object

Exceptions

CMSException

```
6.51.2.34  template<typename T> virtual void
             activemq::commands::ActiveMQMessageTemplate< T
             >::setCMSExpiration ( long long expireTime ) throw (
             cms::CMSException ) [inline, virtual]
```

Sets the Expiration Time for this message.

Parameters

expireTime - time value

Exceptions

CMSException

```
6.51.2.35  template<typename T> virtual void
             activemq::commands::ActiveMQMessageTemplate< T
             >::setCMSMessageID ( const std::string &id AMQCPP_UNUSED )
             throw ( cms::CMSException ) [inline, virtual]
```

Sets the CMS **Message** (p. 2358) Id for this message.

Parameters

id - time value

Exceptions

CMSException


```
6.51.2.36  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSPriority ( int priority ) throw ( cms::CMSEException )
           [inline, virtual]
```

Sets the Priority Value for this message.

Parameters

priority - priority value for this message

Exceptions

CMSEException

```
6.51.2.37  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSRedelivered ( bool redelivered AMQCPP_UNUSED )
           throw ( cms::CMSEException ) [inline, virtual]
```

Sets the Redelivered Flag for this message.

Parameters

redelivered - boolean redelivered value

Exceptions

CMSEException

```
6.51.2.38  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSReplyTo ( const cms::Destination * destination ) throw (
           cms::CMSEException ) [inline, virtual]
```

Sets the CMS Reply To Address for this message.

Parameters

destination Pointer to the CMS Destination that is the Reply-To value.

Exceptions

CMSEException

```
6.51.2.39  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSTimestamp ( long long timeStamp ) throw (
           cms::CMSEException ) [inline, virtual]
```

Sets the Time Stamp for this message.

Parameters

timeStamp - integer time stamp value

Exceptions

CMSException

```
6.51.2.40  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSType ( const std::string & type ) throw ( cms::CMSException
           ) [inline, virtual]
```

Sets the CMS **Message** (p. 2358) Type for this message.

Parameters

type - message type value string

Exceptions

CMSException

```
6.51.2.41  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setDoubleProperty ( const std::string & name, double value )
           throw ( cms::MessageNotWriteableException, cms::CMSException )
           [inline, virtual]
```

Sets a double property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

```
6.51.2.42  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setFloatProperty ( const std::string & name, float value ) throw (
           cms::MessageNotWriteableException, cms::CMSException ) [inline,
           virtual]
```

Sets a float property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.51.2.43 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setIntProperty (const std::string & name, int value) throw (cms::MessageNotWriteableException, cms::CMSException) [inline, virtual]`

Sets a int property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.51.2.44 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setLongProperty (const std::string & name, long long value)
throw (cms::MessageNotWriteableException, cms::CMSException)
[inline, virtual]`

Sets a long property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.51.2.45 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setShortProperty (const std::string & name, short value) throw (cms::MessageNotWriteableException, cms::CMSException) [inline, virtual]`

Sets a short property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

```
6.51.2.46  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setStringProperty (  const std::string &  name,  const
           std::string &  value  ) throw ( cms::MessageNotWriteableException,
           cms::CMSEException ) [inline, virtual]
```

Sets a string property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMessageTemplate.h**

6.52 activemq::commands::ActiveMQObjectMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQObjectMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQObjectMessage:

Public Member Functions

- **ActiveMQObjectMessage** ()
- virtual **~ActiveMQObjectMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQObjectMessage * cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQOBJECTMESSAGE** = 26

6.52.1 Constructor & Destructor Documentation

6.52.1.1 **activemq::commands::ActiveMQObjectMessage::ActiveMQObjectMessage**
()

6.52.1.2 **virtual**
activemq::commands::ActiveMQObjectMessage::~~ActiveMQObjectMessage
() [inline, virtual]

6.52.2 Member Function Documentation

6.52.2.1 **virtual cms::Message*** **activemq::commands::ActiveMQObjectMessage::clone** ()
const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.52.2.2 **virtual ActiveMQObjectMessage*** **activemq::commands::ActiveMQObjectMessage::cloneDataStructure** ()
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2363).

6.52.2.3 `virtual void activemq::commands::ActiveMQObjectMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

`src` - Source Object

Reimplemented from `activemq::commands::Message` (p. 2363).

6.52.2.4 `virtual bool activemq::commands::ActiveMQObjectMessage::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 383).

6.52.2.5 `virtual unsigned char activemq::commands::ActiveMQObjectMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1553) type copy.

Reimplemented from `activemq::commands::Message` (p. 2365).

6.52.2.6 `virtual std::string activemq::commands::ActiveMQObjectMessage::toString () const [virtual]`

Returns a string containing the information for this `DataStructure` (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2372).

6.52.3 Field Documentation

- 6.52.3.1 `const unsigned char activemq::commands::ActiveMQObjectMessage::ID _ - ACTIVEMQOBJECTMESSAGE = 26` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQObjectMessage.h`

6.53 `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 399).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`:

Public Member Functions

- `ActiveMQObjectMessageMarshaller ()`
- `virtual ~ActiveMQObjectMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.53.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 399).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.53.2 Constructor & Destructor Documentation

6.53.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller () [inline]`

6.53.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::~ActiveMQObjectMessageMarshaller () [inline, virtual]`

6.53.3 Member Function Documentation

6.53.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.53.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).


```

6.53.3.3 virtual void ac-
          tivemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseMars
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
            decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

```

6.53.3.4 virtual void ac-
          tivemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseUnmars
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, decaf::io::DataInputStream * dataIn ) throw (
            decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

```

6.53.3.5 virtual int ac-
          tivemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMars
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
            ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2530).

```
6.53.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2531).

```
6.53.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2531).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h`

6.54 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 403).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller`:

Public Member Functions

- `ActiveMQObjectMessageMarshaller ()`
- `virtual ~ActiveMQObjectMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.54.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 403).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.54.2 Constructor & Destructor Documentation

6.54.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller () [inline]`

6.54.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::~ActiveMQObjectMessageMarshaller () [inline, virtual]`

6.54.3 Member Function Documentation

6.54.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.54.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```

6.54.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

```

6.54.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseUnmars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

```

6.54.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2542).

```
6.54.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543).

```
6.54.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h`

6.55 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 407).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`:

Public Member Functions

- `ActiveMQObjectMessageMarshaller ()`
- `virtual ~ActiveMQObjectMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.55.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 407).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.55.2 Constructor & Destructor Documentation

6.55.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller () [inline]`

6.55.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::~ActiveMQObjectMessageMarshaller () [inline, virtual]`

6.55.3 Member Function Documentation

6.55.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.55.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).


```

6.55.3.3 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::looseMars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
          decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

```

6.55.3.4 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::looseUnmars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, decaf::io::DataInputStream * dataIn ) throw (
          decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

```

6.55.3.5 virtual int ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
        ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2538).

```
6.55.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539).

```
6.55.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h`

6.56 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 411).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`:

Public Member Functions

- `ActiveMQObjectMessageMarshaller ()`
- `virtual ~ActiveMQObjectMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.56.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 411).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.56.2 Constructor & Destructor Documentation

6.56.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller () [inline]`

6.56.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::~ActiveMQObjectMessageMarshaller () [inline, virtual]`

6.56.3 Member Function Documentation

6.56.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.56.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```

6.56.3.3 virtual void ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::looseMars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
          decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2524).

```

6.56.3.4 virtual void ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::looseUnmars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, decaf::io::DataInputStream * dataIn ) throw (
          decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2525).

```

6.56.3.5 virtual int ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
        ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

```
6.56.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

```
6.56.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2527).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h`

6.57 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 415).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`:

Public Member Functions

- `ActiveMQObjectMessageMarshaller ()`
- `virtual ~ActiveMQObjectMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.57.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 415).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.57.2 Constructor & Destructor Documentation

6.57.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller () [inline]`

6.57.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::~ActiveMQObjectMessageMarshaller () [inline, virtual]`

6.57.3 Member Function Documentation

6.57.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.57.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).


```

6.57.3.3 virtual void ac-
        tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseMars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
          decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

```

6.57.3.4 virtual void ac-
        tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseUnmars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, decaf::io::DataInputStream * dataIn ) throw (
          decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

```

6.57.3.5 virtual int ac-
        tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
        ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2534).

```
6.57.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535).

```
6.57.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h`

6.58 activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 419).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`:

Public Member Functions

- `ActiveMQObjectMessageMarshaller ()`
- `virtual ~ActiveMQObjectMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.58.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 419).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.58.2 Constructor & Destructor Documentation

6.58.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller () [inline]`

6.58.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::~ActiveMQObjectMessageMarshaller () [inline, virtual]`

6.58.3 Member Function Documentation

6.58.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.58.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```

6.58.3.3 virtual void ac-
        tivemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::looseMars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
          decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

```

6.58.3.4 virtual void ac-
        tivemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::looseUnmars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, decaf::io::DataInputStream * dataIn ) throw (
          decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

```

6.58.3.5 virtual int ac-
        tivemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightMars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
          ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2546).

```
6.58.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2547).

```
6.58.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2547).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMarshaller.h`

6.59 activemq::core::ActiveMQProducer Class Reference

```
#include <src/main/activemq/core/ActiveMQProducer.h>
```

Inheritance diagram for **activemq::core::ActiveMQProducer**:

Public Member Functions

- **ActiveMQProducer** (**ActiveMQSession** *session, const **Pointer**<**commands::ProducerId** > &producerId, const **Pointer**<**commands::ActiveMQDestination** > &destination, long long sendTimeout)

*Constructor, creates an instance of an **ActiveMQProducer** (p. 423).*

- virtual **~ActiveMQProducer** ()
- virtual void **close** () throw (cms::CMSException)

Closes the Consumer.

- virtual void **send** (cms::Message *message) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (cms::Message *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const cms::Destination *destination, cms::Message *message) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const cms::Destination *destination, cms::Message *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **setDeliveryMode** (int mode) throw (cms::CMSEException)
Sets the delivery mode for this Producer.
- virtual int **getDeliveryMode** () const throw (cms::CMSEException)
Gets the delivery mode for this Producer.
- virtual void **setDisableMessageID** (bool value) throw (cms::CMSEException)
Sets if Message Ids are disabled for this Producer.
- virtual bool **getDisableMessageID** () const throw (cms::CMSEException)
Gets if Message Ids are disabled for this Producer.
- virtual void **setDisableMessageTimeStamp** (bool value) throw (cms::CMSEException)
Sets if Message Time Stamps are disabled for this Producer.
- virtual bool **getDisableMessageTimeStamp** () const throw (cms::CMSEException)
Gets if Message Time Stamps are disabled for this Producer.
- virtual void **setPriority** (int priority) throw (cms::CMSEException)
Sets the Priority that this Producers sends messages at.
- virtual int **getPriority** () const throw (cms::CMSEException)
Gets the Priority level that this producer sends messages at.
- virtual void **setTimeToLive** (long long time) throw (cms::CMSEException)
Sets the Time to Live that this Producers sends messages with.
- virtual long long **getTimeToLive** () const throw (cms::CMSEException)
Gets the Time to Live that this producer sends messages with.
- virtual void **setSendTimeout** (long long time) throw (cms::CMSEException)
Sets the Send Timeout that this Producers sends messages with.
- virtual long long **getSendTimeout** () const throw (cms::CMSEException)
Gets the Send Timeout that this producer sends messages with.
- bool **isClosed** () const
- const **Pointer**< **commands::ProducerInfo** > & **getProducerInfo** () const
Retrieves this object ProducerInfo pointer.
- const **Pointer**< **commands::ProducerId** > & **getProducerId** () const
Retrieves this object ProducerId or NULL if closed.
- virtual void **onProducerAck** (const **commands::ProducerAck** &ack)
Handles the work of Processing a ProducerAck Command from the Broker.

6.59.1 Constructor & Destructor Documentation

6.59.1.1 `activemq::core::ActiveMQProducer::ActiveMQProducer (ActiveMQSession * session, const Pointer< commands::ProducerId > & producerId, const Pointer< commands::ActiveMQDestination > & destination, long long sendTimeout)`

Constructor, creates an instance of an **ActiveMQProducer** (p. 423).

Parameters

session The Session which is the parent of this Producer.

producerId Pointer to a ProducerId object which identifies this producer.

destination The assigned Destination this Producer sends to, or null if not set. The Producer does not own the Pointer passed.

sendTimeout The configured send timeout for this Producer.

6.59.1.2 `virtual activemq::core::ActiveMQProducer::~~ActiveMQProducer ()`
[virtual]

6.59.2 Member Function Documentation

6.59.2.1 `virtual void activemq::core::ActiveMQProducer::close () throw (cms::CMSException)` [virtual]

Closes the Consumer.

This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

Exceptions

CMSException

Implements **cms::Closeable** (p. 1065).

6.59.2.2 `virtual int activemq::core::ActiveMQProducer::getDeliveryMode () const throw (cms::CMSException)` [inline, virtual]

Gets the delivery mode for this Producer.

Returns

The DeliveryMode

Implements **cms::MessageProducer** (p. 2552).

6.59.2.3 `virtual bool activemq::core::ActiveMQProducer::getDisableMessageID () const throw (cms::CMSException)` [inline, virtual]

Gets if Message Ids are disabled for this Producer.

Returns

a boolean indicating state enable / disable (true / false) for MessageIds.

Implements **cms::MessageProducer** (p. 2552).

6.59.2.4 `virtual bool activemq::core::ActiveMQProducer::getDisableMessageTimeStamp () const throw (cms::CMSException) [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

Returns

boolean indicating state of enable / disable (true / false)

Implements **cms::MessageProducer** (p. 2553).

6.59.2.5 `virtual int activemq::core::ActiveMQProducer::getPriority () const throw (cms::CMSException) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Implements **cms::MessageProducer** (p. 2553).

6.59.2.6 `const Pointer<commands::ProducerId>& activemq::core::ActiveMQProducer::getProducerId () const [inline]`

Retries this object ProducerId or NULL if closed.

Returns

ProducerId Reference

6.59.2.7 `const Pointer<commands::ProducerInfo>& activemq::core::ActiveMQProducer::getProducerInfo () const [inline]`

Retries this object ProducerInfo pointer.

Returns

ProducerInfo Reference

6.59.2.8 `virtual long long activemq::core::ActiveMQProducer::getSendTimeout () const throw (cms::CMSException) [inline, virtual]`

Gets the Send Timeout that this producer sends messages with.

Returns

The default send timeout value in milliseconds.

6.59.2.9 `virtual long long activemq::core::ActiveMQProducer::getTimeToLive () const throw (cms::CMSException) [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

Returns

The default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 2553).

6.59.2.10 `bool activemq::core::ActiveMQProducer::isClosed () const [inline]`

Returns

true if this Producer has been closed.

6.59.2.11 `virtual void activemq::core::ActiveMQProducer::onProducerAck (const commands::ProducerAck & ack) [virtual]`

Handles the work of Processing a ProducerAck Command from the Broker.

Parameters

ack - The ProducerAck message received from the Broker.

6.59.2.12 `virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * destination, cms::Message * message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters

destination The destination on which to send the message

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSEException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2554).

6.59.2.13 `virtual void activemq::core::ActiveMQProducer::send (cms::Message * message) throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

message The message to be sent.

Exceptions

CMSEException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2555).

6.59.2.14 `virtual void activemq::core::ActiveMQProducer::send (cms::Message * message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2553).

6.59.2.15 `virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * destination, cms::Message * message) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

destination The destination on which to send the message

message the message to be sent.

Exceptions

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2555).

6.59.2.16 `virtual void activemq::core::ActiveMQProducer::setDeliveryMode (int mode) throw (cms::CMSException) [inline, virtual]`

Sets the delivery mode for this Producer.

Parameters

mode - The DeliveryMode to use for Message sends.

Implements **cms::MessageProducer** (p. 2556).

6.59.2.17 `virtual void activemq::core::ActiveMQProducer::setDisableMessageID (bool value) throw (cms::CMSException) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

Parameters

value - boolean indicating enable / disable (true / false)

Implements `cms::MessageProducer` (p. 2556).

6.59.2.18 `virtual void activemq::core::ActiveMQProducer::setDisableMessageTimeStamp (bool value) throw (cms::CMSException) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

Parameters

value - boolean indicating enable / disable (true / false)

Implements `cms::MessageProducer` (p. 2556).

6.59.2.19 `virtual void activemq::core::ActiveMQProducer::setPriority (int priority) throw (cms::CMSException) [inline, virtual]`

Sets the Priority that this Producers sends messages at.

Parameters

priority int value for Priority level

Implements `cms::MessageProducer` (p. 2556).

6.59.2.20 `virtual void activemq::core::ActiveMQProducer::setSendTimeout (long long time) throw (cms::CMSException) [inline, virtual]`

Sets the Send Timeout that this Producers sends messages with.

Parameters

time The new default send timeout value in milliseconds.

6.59.2.21 `virtual void activemq::core::ActiveMQProducer::setTimeToLive (long long time) throw (cms::CMSException) [inline, virtual]`

Sets the Time to Live that this Producers sends messages with.

Parameters

time The new default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 2557).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQProducer.h`

6.60 activemq::util::ActiveMQProperties Class Reference

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2927) object.

```
#include <src/main/activemq/util/ActiveMQProperties.h>
```

Inheritance diagram for `activemq::util::ActiveMQProperties`:

Public Member Functions

- **ActiveMQProperties** ()
- virtual **~ActiveMQProperties** ()
- virtual **decaf::util::Properties** & **getProperties** ()
- virtual const **decaf::util::Properties** & **getProperties** () const
- virtual void **setProperties** (**decaf::util::Properties** &props)
- virtual bool **isEmpty** () const
Returns true if the properties object is empty.
- virtual const char * **getProperty** (const std::string &name) const
Looks up the value for the given property.
- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const
Looks up the value for the given property.
- virtual void **setProperty** (const std::string &name, const std::string &value)
Sets the value for a given property.
- virtual bool **hasProperty** (const std::string &name) const
Check to see if the Property exists in the set.
- virtual void **remove** (const std::string &name)
Removes the property with the given name.
- virtual std::vector< std::pair< std::string, std::string > > **toArray** () const
Method that serializes the contents of the property map to an array.
- virtual void **copy** (const CMSProperties *source)
Copies the contents of the given properties object to this one.
- virtual CMSProperties * **clone** () const
Clones this object.

- virtual void **clear** ()
Clears all properties from the map.
- virtual std::string **toString** () const
Formats the contents of the Properties Object into a string that can be logged, etc.

6.60.1 Detailed Description

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2927) object.

Since

2.0

6.60.2 Constructor & Destructor Documentation

6.60.2.1 **activemq::util::ActiveMQProperties::ActiveMQProperties** ()

6.60.2.2 **virtual activemq::util::ActiveMQProperties::~~ActiveMQProperties** ()
[virtual]

6.60.3 Member Function Documentation

6.60.3.1 **virtual void activemq::util::ActiveMQProperties::clear** () [inline, virtual]

Clears all properties from the map.

Implements **cms::CMSProperties** (p. 1080).

6.60.3.2 **virtual CMSProperties* activemq::util::ActiveMQProperties::clone** ()
const [virtual]

Clones this object.

Returns

a replica of this object.

Implements **cms::CMSProperties** (p. 1080).

6.60.3.3 **virtual void activemq::util::ActiveMQProperties::copy** (**const CMSProperties * source**) [virtual]

Copies the contents of the given properties object to this one.

Parameters

source The source properties object.

6.60.3.4 `virtual decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ()`
[inline, virtual]

6.60.3.5 `virtual const decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ()`
`const` [inline, virtual]

6.60.3.6 `virtual const char* activemq::util::ActiveMQProperties::getProperty (`
`const std::string & name) const` [inline, virtual]

Looks up the value for the given property.

Parameters

name The name of the property to be looked up.

Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implements `cms::CMSProperties` (p. 1080).

6.60.3.7 `virtual std::string activemq::util::ActiveMQProperties::getProperty (`
`const std::string & name, const std::string & defaultValue) const`
[inline, virtual]

Looks up the value for the given property.

Parameters

name the name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implements `cms::CMSProperties` (p. 1081).

6.60.3.8 `virtual bool activemq::util::ActiveMQProperties::hasProperty (const`
`std::string & name) const` [inline, virtual]

Check to see if the Property exists in the set.

Parameters

name - property name to check for in this properties set.

Returns

true if property exists, false otherwise.

Implements `cms::CMSProperties` (p. 1081).

6.60.3.9 `virtual bool activemq::util::ActiveMQProperties::isEmpty () const`
[inline, virtual]

Returns true if the properties object is empty.

Returns

true if empty

Implements `cms::CMSProperties` (p. 1081).

6.60.3.10 `virtual void activemq::util::ActiveMQProperties::remove (const`
`std::string & name)` [inline, virtual]

Removes the property with the given name.

Parameters

name the name of the property to remove.

Implements `cms::CMSProperties` (p. 1081).

6.60.3.11 `virtual void activemq::util::ActiveMQProperties::setProperties (`
`decaf::util::Properties & props)` [inline, virtual]

6.60.3.12 `virtual void activemq::util::ActiveMQProperties::setProperty (const`
`std::string & name, const std::string & value)` [inline, virtual]

Sets the value for a given property.

If the property already exists, overwrites the value.

Parameters

name The name of the value to be written.

value The value to be written.

Implements `cms::CMSProperties` (p. 1082).

6.60.3.13 `virtual std::vector< std::pair< std::string, std::string > >`
`activemq::util::ActiveMQProperties::toArray () const` [inline,
virtual]

Method that serializes the contents of the property map to an array.

Returns

list of pairs where the first is the name and the second is the value.

Implements `cms::CMSProperties` (p. 1082).

6.60.3.14 `virtual std::string activemq::util::ActiveMQProperties::toString ()`
`const [inline, virtual]`

Formats the contents of the Properties Object into a string that can be logged, etc.

Returns

string value of this object.

Implements **cms::CMSProperties** (p. 1082).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/ActiveMQProperties.h`

6.61 activemq::commands::ActiveMQQueue Class Reference

```
#include <src/main/activemq/commands/ActiveMQQueue.h>
```

Inheritance diagram for `activemq::commands::ActiveMQQueue`:

Public Member Functions

- **ActiveMQQueue** ()
- **ActiveMQQueue** (const std::string &name)
- virtual **~ActiveMQQueue** ()
- virtual unsigned char **getDataStructureType** () const
- virtual **ActiveMQQueue * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const
Retrieve the Destination Type for this Destination.
- virtual **cms::Destination** * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)

Copies the contents of the given Destination object to this one.

- virtual const **cms::CMSProperties** & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual std::string **getQueueName** () const throw (cms::CMSException)
Gets the name of this queue.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQQUEUE** = 100

6.61.1 Constructor & Destructor Documentation

- 6.61.1.1 **activemq::commands::ActiveMQQueue::ActiveMQQueue** ()
- 6.61.1.2 **activemq::commands::ActiveMQQueue::ActiveMQQueue** (const std::string & *name*)
- 6.61.1.3 **virtual activemq::commands::ActiveMQQueue::~~ActiveMQQueue** ()
[inline, virtual]

6.61.2 Member Function Documentation

- 6.61.2.1 **virtual cms::Destination* activemq::commands::ActiveMQQueue::clone** () const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p.1612).

- 6.61.2.2 **virtual ActiveMQQueue* activemq::commands::ActiveMQQueue::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

- 6.61.2.3 **virtual void activemq::commands::ActiveMQQueue::copy** (const cms::Destination & *source*) [inline, virtual]

Copies the contents of the given Destination object to this one.

Parameters

source The source Destination object.

Implements **cms::Destination** (p.1612).

6.61.2.4 virtual void activemq::commands::ActiveMQQueue::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

6.61.2.5 virtual bool activemq::commands::ActiveMQQueue::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p.1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.61.2.6 virtual const cms::Destination* activemq::commands::ActiveMQQueue::getCMSDestination () const [inline, virtual]

Returns

the **cms::Destination** (p.1610) interface pointer that the objects that derive from this class implement.

6.61.2.7 virtual const cms::CMSProperties& activemq::commands::ActiveMQQueue::getCMSProperties () const [inline, virtual]

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

const reference to a properties object.

Implements **cms::Destination** (p.1612).

6.61.2.8 `virtual unsigned char activemq::commands::ActiveMQQueue::getDataStructureType () const`
[virtual]

6.61.2.9 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQQueue::getDestinationType () const`
[inline, virtual]

Retrieve the Destination Type for this Destination.

Returns

The Destination Type

Implements **cms::Destination** (p. 1612).

6.61.2.10 `virtual std::string activemq::commands::ActiveMQQueue::getQueueName () const throw (cms::CMSException)` [inline, virtual]

Gets the name of this queue.

Returns

The queue name.

Implements **cms::Queue** (p. 2947).

6.61.2.11 `virtual std::string activemq::commands::ActiveMQQueue::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.61.3 Field Documentation

6.61.3.1 `const unsigned char activemq::commands::ActiveMQQueue::ID _ - ACTIVEMQQUEUE = 100` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQQueue.h`

6.62 activemq::core::ActiveMQQueueBrowser Class Reference

```
#include <src/main/activemq/core/ActiveMQQueueBrowser.h>
```

Inheritance diagram for activemq::core::ActiveMQQueueBrowser:

Public Member Functions

- **ActiveMQQueueBrowser** (**ActiveMQSession** *session, const **Pointer**< **commands::ConsumerId** > &consumerId, const **Pointer**< **commands::ActiveMQDestination** > &destination, const std::string &selector, bool dispatchAsync)
- virtual ~**ActiveMQQueueBrowser** ()
- virtual const **cms::Queue** * **getQueue** () const throw (cms::CMSEException)
- virtual std::string **getMessageSelector** () const throw (cms::CMSEException)
- virtual **cms::MessageEnumeration** * **getEnumeration** () throw (cms::CMSEException)
- virtual void **close** () throw (cms::CMSEException)
- virtual bool **hasMoreMessages** ()

*Returns true if there are more Message in the Browser that can be retrieved via the **nextMessage** method.*

- virtual **cms::Message** * **nextMessage** () throw (cms::CMSEException)

Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown.

Friends

- class **Browser**

6.62.1 Constructor & Destructor Documentation

- 6.62.1.1** `activemq::core::ActiveMQQueueBrowser::ActiveMQQueueBrowser (ActiveMQSession * session, const Pointer< commands::ConsumerId > & consumerId, const Pointer< commands::ActiveMQDestination > & destination, const std::string & selector, bool dispatchAsync)`
- 6.62.1.2** `virtual
activemq::core::ActiveMQQueueBrowser::~~ActiveMQQueueBrowser ()
[virtual]`

6.62.2 Member Function Documentation

- 6.62.2.1** `virtual void activemq::core::ActiveMQQueueBrowser::close () throw (cms::CMSException) [virtual]`
- 6.62.2.2** `virtual cms::MessageEnumeration* activemq::core::ActiveMQQueueBrowser::getEnumeration () throw (cms::CMSException) [virtual]`
- 6.62.2.3** `virtual std::string activemq::core::ActiveMQQueueBrowser::getMessageSelector () const throw (cms::CMSException) [virtual]`
- 6.62.2.4** `virtual const cms::Queue* activemq::core::ActiveMQQueueBrowser::getQueue () const throw (cms::CMSException) [virtual]`
- 6.62.2.5** `virtual bool activemq::core::ActiveMQQueueBrowser::hasMoreMessages () [virtual]`

Returns true if there are more Message in the Browser that can be retrieved via the `nextMessage` method.

If this method returns false and the `nextMessage` method is called then an Exception will be thrown.

Returns

true if more Message's are available in the Browser.

Implements `cms::MessageEnumeration` (p. 2491).

- 6.62.2.6** `virtual cms::Message* activemq::core::ActiveMQQueueBrowser::nextMessage () throw (cms::CMSException) [virtual]`

Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown.

If a Message object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

Returns

The next Message in the Queue.

Exceptions

CMSException if no more Message's currently in the Queue.

Implements `cms::MessageEnumeration` (p. 2491).

6.62.3 Friends And Related Function Documentation

6.62.3.1 friend class Browser [friend]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQQueueBrowser.h`

6.63 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQQueueMarshaller` (p. 441).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller`:

Public Member Functions

- `ActiveMQQueueMarshaller ()`
- `virtual ~ActiveMQQueueMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.63.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 441). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.63.2 Constructor & Destructor Documentation

6.63.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller () [inline]`

6.63.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller () [inline, virtual]`

6.63.3 Member Function Documentation

6.63.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.63.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.63.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 291).

6.63.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 291).

6.63.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 292).

```
6.63.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 292).

```
6.63.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 293).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h`

6.64 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 445).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller**:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.64.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 445). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.64.2 Constructor & Destructor Documentation

6.64.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller () [inline]`

6.64.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller () [inline, virtual]`

6.64.3 Member Function Documentation

6.64.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.64.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.64.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 295).

6.64.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 295).

6.64.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 296).

```
6.64.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 296).

```
6.64.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 297).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h`

6.65 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 449).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller**:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.65.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 449). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.65.2 Constructor & Destructor Documentation

6.65.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller () [inline]`

6.65.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller () [inline, virtual]`

6.65.3 Member Function Documentation

6.65.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.65.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.65.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 299).

6.65.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 299).

6.65.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 300).

```
6.65.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 300).

```
6.65.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 301).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h`

6.66 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 453).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller**:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.66.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 453). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.66.2 Constructor & Destructor Documentation

6.66.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller () [inline]`

6.66.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller () [inline, virtual]`

6.66.3 Member Function Documentation

6.66.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.66.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.66.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 303).

6.66.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 303).

6.66.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 304).

```
6.66.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 304).

```
6.66.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 305).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h`

6.67 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 457).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller**:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.67.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 457). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.67.2 Constructor & Destructor Documentation

6.67.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller () [inline]`

6.67.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller () [inline, virtual]`

6.67.3 Member Function Documentation

6.67.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.67.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.67.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 307).

6.67.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 307).

6.67.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 308).

```
6.67.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 308).

```
6.67.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 309).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h`

6.68 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 461).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller**:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.68.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 461). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.68.2 Constructor & Destructor Documentation

6.68.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller () [inline]`

6.68.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller () [inline, virtual]`

6.68.3 Member Function Documentation

6.68.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.68.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.68.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 311).

6.68.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 311).

6.68.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 312).

```
6.68.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 312).

```
6.68.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 313).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h`

6.69 activemq::core::ActiveMQSession Class Reference

```
#include <src/main/activemq/core/ActiveMQSession.h>
```

Inheritance diagram for **activemq::core::ActiveMQSession**:

Public Member Functions

- **ActiveMQSession** (const **Pointer**< **commands::SessionInfo** > &sessionInfo, **cms::Session::AcknowledgeMode** ackMode, const **decaf::util::Properties** &properties, **ActiveMQConnection** *connection)
- virtual **~ActiveMQSession** ()
- void **redispatch** (**MessageDispatchChannel** &unconsumedMessages)
Redispatches the given set of unconsumed messages to the consumers.
- void **start** ()
Stops asynchronous message delivery.
- void **stop** ()
Starts asynchronous message delivery.
- bool **isStarted** () const
Indicates whether or not the session is currently in the started state.
- bool **isAutoAcknowledge** () const
- bool **isDupsOkAcknowledge** () const
- bool **isClientAcknowledge** () const
- bool **isIndividualAcknowledge** () const
- void **fire** (const **exceptions::ActiveMQException** &ex)
Fires the given exception to the exception listener of the connection.
- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)
Dispatches a message to a particular consumer.
- virtual void **close** () throw (**cms::CMSEException**)
Closes this session as well as any active child consumers or producers.
- virtual void **commit** () throw (**cms::CMSEException**)

Commits all messages done in this transaction and releases any locks currently held.

- virtual void **rollback** () throw (cms::CMSEException)
Rollsback all messages done in this transaction and releases any locks currently held.
- virtual void **recover** () throw (cms::CMSEException)
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual cms::MessageConsumer * **createConsumer** (const cms::Destination *destination) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination.
- virtual cms::MessageConsumer * **createConsumer** (const cms::Destination *destination, const std::string &selector) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual cms::MessageConsumer * **createConsumer** (const cms::Destination *destination, const std::string &selector, bool noLocal) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual cms::MessageConsumer * **createDurableConsumer** (const cms::Topic *destination, const std::string &name, const std::string &selector, bool noLocal=false) throw (cms::CMSEException)
Creates a durable subscriber to the specified topic, using a message selector.
- virtual cms::MessageProducer * **createProducer** (const cms::Destination *destination) throw (cms::CMSEException)
Creates a MessageProducer to send messages to the specified destination.
- virtual cms::QueueBrowser * **createBrowser** (const cms::Queue *queue) throw (cms::CMSEException)
Creates a new QueueBrowser to peek at Messages on the given Queue.
- virtual cms::QueueBrowser * **createBrowser** (const cms::Queue *queue, const std::string &selector) throw (cms::CMSEException)
Creates a new QueueBrowser to peek at Messages on the given Queue.
- virtual cms::Queue * **createQueue** (const std::string &queueName) throw (cms::CMSEException)
Creates a queue identity given a Queue name.
- virtual cms::Topic * **createTopic** (const std::string &topicName) throw (cms::CMSEException)
Creates a topic identity given a Queue name.
- virtual cms::TemporaryQueue * **createTemporaryQueue** () throw (cms::CMSEException)
Creates a TemporaryQueue object.

- virtual **cms::TemporaryTopic * createTemporaryTopic ()** throw (cms::CMSEException)
Creates a TemporaryTopic object.
- virtual **cms::Message * createMessage ()** throw (cms::CMSEException)
Creates a new Message.
- virtual **cms::BytesMessage * createBytesMessage ()** throw (cms::CMSEException)
Creates a BytesMessage.
- virtual **cms::BytesMessage * createBytesMessage (const unsigned char *bytes, int bytesSize)** throw (cms::CMSEException)
Creates a BytesMessage and sets the pay-load to the passed value.
- virtual **cms::StreamMessage * createStreamMessage ()** throw (cms::CMSEException)
Creates a new StreamMessage.
- virtual **cms::TextMessage * createTextMessage ()** throw (cms::CMSEException)
Creates a new TextMessage.
- virtual **cms::TextMessage * createTextMessage (const std::string &text)** throw (cms::CMSEException)
Creates a new TextMessage and set the text to the value given.
- virtual **cms::MapMessage * createMapMessage ()** throw (cms::CMSEException)
Creates a new MapMessage.
- virtual **cms::Session::AcknowledgeMode getAcknowledgeMode ()** const throw (cms::CMSEException)
Returns the acknowledgment mode of the session.
- virtual **bool isTransacted ()** const throw (cms::CMSEException)
Gets if the Sessions is a Transacted Session.
- virtual **void unsubscribe (const std::string &name)** throw (cms::CMSEException)
Unsubscribes a durable subscription that has been created by a client.
- **void send (cms::Message *message, ActiveMQProducer *producer, util::Usage *usage)** throw (cms::CMSEException)
Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.
- **cms::ExceptionListener * getExceptionListener ()**
This method gets any registered exception listener of this sessions connection and returns it.
- **const commands::SessionInfo & getSessionInfo ()** const
Gets the Session Information object for this session, if the session is closed than this method throws an exception.
- **const commands::SessionId & getSessionId ()** const

Gets the Session Id object for this session, if the session is closed than this method throws an exception.

- **ActiveMQConnection * getConnection () const**
*Gets the **ActiveMQConnection** (p. 233) that is associated with this session.*
- **long long getLastDeliveredSequenceId () const**
Gets the currently set Last Delivered Sequence Id.
- **void setLastDeliveredSequenceId (long long value)**
Sets the value of the Last Delivered Sequence Id.
- **void oneway (Pointer< commands::Command > command) throw (activemq::exceptions::ActiveMQException)**
Sends a oneway message.
- **void syncRequest (Pointer< commands::Command > command, unsigned int timeout=0) throw (activemq::exceptions::ActiveMQException)**
Sends a synchronous request and returns the response from the broker.
- **void addConsumer (ActiveMQConsumer *consumer) throw (activemq::exceptions::ActiveMQException)**
Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.
- **void removeConsumer (const Pointer< commands::ConsumerId > &consumerId, long long lastDeliveredSequenceId=0) throw (activemq::exceptions::ActiveMQException)**
Dispose of a MessageConsumer from this session.
- **void addProducer (ActiveMQProducer *consumer) throw (activemq::exceptions::ActiveMQException)**
Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.
- **void removeProducer (const Pointer< commands::ProducerId > &producerId) throw (activemq::exceptions::ActiveMQException)**
Dispose of a MessageProducer from this session.
- **void doStartTransaction () throw (exceptions::ActiveMQException)**
Starts if not already start a Transaction for this Session.
- **Pointer< ActiveMQTransactionContext > getTransactionContext ()**
Gets the Pointer to this Session's TransactionContext.
- **void acknowledge ()**
Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.
- **void deliverAcks ()**
Request that this Session inform all of its consumers to deliver their pending acks.

- void **clearMessagesInProgress** ()
Request that this Session inform all of its consumers to clear all messages that are currently in progress.
- void **wakeup** ()
Causes the Session to wakeup its executor and ensure all messages are dispatched.
- **Pointer< commands::ConsumerId > getNextConsumerId** ()
Get the Next available Consumer Id.
- **Pointer< commands::ProducerId > getNextProducerId** ()
Get the Next available Producer Id.

Friends

- class **ActiveMQSessionExecutor**

6.69.1 Constructor & Destructor Documentation

- 6.69.1.1** **activemq::core::ActiveMQSession::ActiveMQSession** (const **Pointer< commands::SessionInfo >** & *sessionInfo*, **cms::Session::AcknowledgeMode** *ackMode*, const **decaf::util::Properties** & *properties*, **ActiveMQConnection** * *connection*)
- 6.69.1.2** **virtual activemq::core::ActiveMQSession::~~ActiveMQSession** ()
[virtual]

6.69.2 Member Function Documentation

- 6.69.2.1** **void activemq::core::ActiveMQSession::acknowledge** ()

Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.

- 6.69.2.2** **void activemq::core::ActiveMQSession::addConsumer**
(**ActiveMQConsumer** * *consumer*) throw (**activemq::exceptions::ActiveMQException**)

Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.

Parameters

consumer The **ActiveMQConsumer** (p. 268) instance to add to this session.

Exceptions

ActiveMQException if an internal error occurs.

6.69.2.3 `void activemq::core::ActiveMQSession::addProducer (ActiveMQProducer * consumer) throw (activemq::exceptions::ActiveMQException)`

Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.

Parameters

consumer The **ActiveMQProducer** (p. 423) instance to add to this session.

Exceptions

ActiveMQException if an internal error occurs.

6.69.2.4 `void activemq::core::ActiveMQSession::clearMessagesInProgress ()`

Request that this Session inform all of its consumers to clear all messages that are currently in progress.

6.69.2.5 `virtual void activemq::core::ActiveMQSession::close () throw (cms::CMSException) [virtual]`

Closes this session as well as any active child consumers or producers.

Exceptions

CMSException

6.69.2.6 `virtual void activemq::core::ActiveMQSession::commit () throw (cms::CMSException) [virtual]`

Commits all messages done in this transaction and releases any locks currently held.

Exceptions

CMSException

6.69.2.7 `virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * queue) throw (cms::CMSException) [virtual]`

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

queue the Queue to browse

Returns

New QueueBrowser that is owned by the caller.

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

6.69.2.8 `virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * queue, const std::string & selector) throw (cms::CMSException) [virtual]`

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

queue the Queue to browse

selector the Message selector to filter which messages are browsed.

Returns

New QueueBrowser that is owned by the caller.

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

6.69.2.9 `virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage () throw (cms::CMSException) [virtual]`

Creates a BytesMessage.

Returns

a newly created BytesMessage.

Exceptions

CMSException

6.69.2.10 `virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage (const unsigned char * bytes, int bytesSize) throw (cms::CMSException) [virtual]`

Creates a BytesMessage and sets the pay-load to the passed value.

Parameters

bytes - an array of bytes to set in the message

bytesSize - the size of the bytes array, or number of bytes to use

Returns

a newly created BytesMessage.

Exceptions

CMSException

6.69.2.11 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal) throw (cms::CMSException) [virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

destination - The Destination that this consumer receiving messages for.

selector - The Message Selector string to use for this destination

noLocal - if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Exceptions

CMSException

6.69.2.12 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination, const std::string & selector) throw (cms::CMSException) [virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

destination - The Destination that this consumer receiving messages for.

selector - The Message Selector string to use for this destination

Exceptions

CMSException

6.69.2.13 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination) throw (cms::CMSException) [virtual]`

Creates a MessageConsumer for the specified destination.

Parameters

destination - The Destination that this consumer receiving messages for.

Exceptions

CMSException

6.69.2.14 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createDurableConsumer (const cms::Topic * *destination*, const std::string & *name*, const std::string & *selector*, bool *noLocal* = *false*) throw (cms::CMSException) [virtual]

Creates a durable subscriber to the specified topic, using a message selector.

Parameters

destination - the topic to subscribe to

name - The name used to identify the subscription

selector - only messages matching the selector are received

noLocal - if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Exceptions

CMSException

6.69.2.15 virtual cms::MapMessage* activemq::core::ActiveMQSession::createMapMessage () throw (cms::CMSException) [virtual]

Creates a new MapMessage.

Returns

a newly created MapMessage.

Exceptions

CMSException

6.69.2.16 virtual cms::Message* activemq::core::ActiveMQSession::createMessage () throw (cms::CMSException) [virtual]

Creates a new Message.

Exceptions

CMSException

6.69.2.17 `virtual cms::MessageProducer* activemq::core::ActiveMQSession::createProducer (const cms::Destination * destination) throw (cms::CMSException)`
[virtual]

Creates a MessageProducer to send messages to the specified destination.

Parameters

destination - the Destination to publish on

Exceptions

CMSException

6.69.2.18 `virtual cms::Queue* activemq::core::ActiveMQSession::createQueue (const std::string & queueName) throw (cms::CMSException)`
[virtual]

Creates a queue identity given a Queue name.

Parameters

queueName - the name of the new Queue

Exceptions

CMSException

6.69.2.19 `virtual cms::StreamMessage* activemq::core::ActiveMQSession::createStreamMessage () throw (cms::CMSException)` [virtual]

Creates a new StreamMessage.

Returns

a newly created StreamMessage.

Exceptions

CMSException

6.69.2.20 `virtual cms::TemporaryQueue* activemq::core::ActiveMQSession::createTemporaryQueue () throw (cms::CMSException)` [virtual]

Creates a TemporaryQueue object.

Exceptions

CMSException

6.69.2.21 virtual cms::TemporaryTopic* activemq::core::ActiveMQSession::createTemporaryTopic () throw (cms::CMSException) [virtual]

Creates a TemporaryTopic object.

Exceptions

CMSException

6.69.2.22 virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage () throw (cms::CMSException) [virtual]

Creates a new TextMessage.

Returns

a newly created TextMessage.

Exceptions

CMSException

6.69.2.23 virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage (const std::string & *text*) throw (cms::CMSException) [virtual]

Creates a new TextMessage and set the text to the value given.

Parameters

text - The initial text for the message

Returns

a newly created TextMessage with the given Text set in the Message body.

Exceptions

CMSException

6.69.2.24 virtual cms::Topic* activemq::core::ActiveMQSession::createTopic (const std::string & *topicName*) throw (cms::CMSException) [virtual]

Creates a topic identity given a Queue name.

Parameters

topicName - the name of the new Topic

Exceptions

CMSException

6.69.2.25 void activemq::core::ActiveMQSession::deliverAcks ()

Request that this Session inform all of its consumers to deliver their pending acks.

6.69.2.26 virtual void activemq::core::ActiveMQSession::dispatch (const Pointer< MessageDispatch > & message) [virtual]

Dispatches a message to a particular consumer.

Parameters

message - the message to be dispatched

Implements **activemq::core::Dispatcher** (p. 1672).

6.69.2.27 void activemq::core::ActiveMQSession::doStartTransaction () throw (exceptions::ActiveMQException)

Starts if not already start a Transaction for this Session.

If the session is not a Transacted Session then an exception is thrown. If a transaction is already in progress then this method has no effect.

Exceptions

ActiveMQException if this is not a Transacted Session.

6.69.2.28 void activemq::core::ActiveMQSession::fire (const exceptions::ActiveMQException & ex)

Fires the given exception to the exception listener of the connection.

6.69.2.29 virtual cms::Session::AcknowledgeMode activemq::core::ActiveMQSession::getAcknowledgeMode () const throw (cms::CMSException) [virtual]

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

6.69.2.30 ActiveMQConnection* activemq::core::ActiveMQSession::getConnection () const [inline]

Gets the **ActiveMQConnection** (p. 233) that is associated with this session.

6.69.2.31 `cms::ExceptionListener* activemq::core::ActiveMQSession::getExceptionListener ()`

This method gets any registered exception listener of this sessions connection and returns it.

Mainly intended for use by the objects that this session creates so that they can notify the client of exceptions that occur in the context of another thread.

Returns

`cms::ExceptionListener` (p. 1719) pointer or NULL

6.69.2.32 `long long activemq::core::ActiveMQSession::getLastDeliveredSequenceId () const [inline]`

Gets the currently set Last Delivered Sequence Id.

Returns

long long containing the sequence id of the last delivered Message.

6.69.2.33 `Pointer<commands::ConsumerId> activemq::core::ActiveMQSession::getNextConsumerId ()`

Get the Next available Consumer Id.

Returns

the next id in the sequence.

6.69.2.34 `Pointer<commands::ProducerId> activemq::core::ActiveMQSession::getNextProducerId ()`

Get the Next available Producer Id.

Returns

the next id in the sequence.

6.69.2.35 `const commands::SessionId& activemq::core::ActiveMQSession::getSessionId () const [inline]`

Gets the Session Id object for this session, if the session is closed than this method throws an exception.

Returns

SessionId Reference

6.69.2.36 `const commands::SessionInfo& activemq::core::ActiveMQSession::getSessionInfo () const`
`[inline]`

Gets the Session Information object for this session, if the session is closed than this method throws an exception.

Returns

SessionInfo Reference

6.69.2.37 `Pointer<ActiveMQTransactionContext> activemq::core::ActiveMQSession::getTransactionContext ()`
`[inline]`

Gets the Pointer to this Session's TransactionContext.

Returns

a Pointer to this Session's TransactionContext

6.69.2.38 `bool activemq::core::ActiveMQSession::isAutoAcknowledge () const`
`[inline]`

6.69.2.39 `bool activemq::core::ActiveMQSession::isClientAcknowledge () const`
`[inline]`

6.69.2.40 `bool activemq::core::ActiveMQSession::isDupsOkAcknowledge () const`
`[inline]`

6.69.2.41 `bool activemq::core::ActiveMQSession::isIndividualAcknowledge ()`
`const [inline]`

6.69.2.42 `bool activemq::core::ActiveMQSession::isStarted () const`

Indicates whether or not the session is currently in the started state.

6.69.2.43 `virtual bool activemq::core::ActiveMQSession::isTransacted () const`
`throw (cms::CMSException) [virtual]`

Gets if the Sessions is a Transacted Session.

Returns

transacted true - false.

6.69.2.44 `void activemq::core::ActiveMQSession::oneway (`
`Pointer< commands::Command > command) throw (`
`activemq::exceptions::ActiveMQException)`

Sends a oneway message.

Parameters

command The message to send.

Exceptions

ActiveMQException if not currently connected, or if the operation fails for any reason.

6.69.2.45 virtual void activemq::core::ActiveMQSession::recover () throw (cms::CMSException) [virtual]

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions

CMSException - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException - if the method is called by a transacted session.

6.69.2.46 void activemq::core::ActiveMQSession::redispatch (MessageDispatchChannel & *unconsumedMessages*)

Redispatches the given set of unconsumed messages to the consumers.

Parameters

unconsumedMessages - unconsumed messages to be redelivered.

6.69.2.47 void activemq::core::ActiveMQSession::removeConsumer (const Pointer< commands::ConsumerId > & *consumerId*, long long *lastDeliveredSequenceId* = 0) throw (activemq::exceptions::ActiveMQException)

Dispose of a MessageConsumer from this session.

Removes it from the Connection and clean up any resources associated with it.

Parameters

consumerId The ConsumerId of the MessageConsumer to remove from this Session.

lastDeliveredSequenceId The sequenceId of the last Message the consumer delivered.

Exceptions

ActiveMQException if an internal error occurs.

6.69.2.48 `void activemq::core::ActiveMQSession::removeProducer (const
Pointer< commands::ProducerId > & producerId) throw (
activemq::exceptions::ActiveMQException)`

Dispose of a MessageProducer from this session.

Removes it from the Connection and clean up any resources associated with it.

Parameters

producerId The ProducerId of the MessageProducer to remove from this session.

Exceptions

ActiveMQException if an internal error occurs.

6.69.2.49 `virtual void activemq::core::ActiveMQSession::rollback () throw (
cms::CMSException) [virtual]`

Rollsback all messages done in this transaction and releases any locks currently held.

Exceptions

CMSException

6.69.2.50 `void activemq::core::ActiveMQSession::send (cms::Message * message,
ActiveMQProducer * producer, util::Usage * usage) throw (
cms::CMSException)`

Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.

Asynchronous sends will be chosen if at all possible.

Parameters

message The message to send to the broker.

producer The sending Producer

usage Pointer to a Usage tracker which if set will be increased by the size of the given message.

Exceptions

CMSException

6.69.2.51 `void activemq::core::ActiveMQSession::setLastDeliveredSequenceId (long long value) [inline]`

Sets the value of the Last Delivered Sequence Id.

Parameters

value The new value to assign to the Last Delivered Sequence Id property.

6.69.2.52 `void activemq::core::ActiveMQSession::start ()`

Stops asynchronous message delivery.

6.69.2.53 `void activemq::core::ActiveMQSession::stop ()`

Starts asynchronous message delivery.

6.69.2.54 `void activemq::core::ActiveMQSession::syncRequest (Pointer< commands::Command > command, unsigned int timeout = 0) throw (activemq::exceptions::ActiveMQException)`

Sends a synchronous request and returns the response from the broker.

Converts any error responses into an exception.

Parameters

command The request command.

timeout The time to wait for a response, default is zero or infinite.

Exceptions

ActiveMQException thrown if an error response was received from the broker, or if any other error occurred.

6.69.2.55 `virtual void activemq::core::ActiveMQSession::unsubscribe (const std::string & name) throw (cms::CMSException) [virtual]`

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters

name the name used to identify this subscription

Exceptions

CMSException

6.69.2.56 void activemq::core::ActiveMQSession::wakeup ()

Causes the Session to wakeup its executor and ensure all messages are dispatched.

6.69.3 Friends And Related Function Documentation

6.69.3.1 friend class ActiveMQSessionExecutor [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQSession.h**

6.70 activemq::core::ActiveMQSessionExecutor Class Reference

Delegate dispatcher for a single session.

```
#include <src/main/activemq/core/ActiveMQSessionExecutor.h>
```

Inheritance diagram for activemq::core::ActiveMQSessionExecutor:

Public Member Functions

- **ActiveMQSessionExecutor** (**ActiveMQSession** *session)
Creates an un-started executor for the given session.
- virtual **~ActiveMQSessionExecutor** ()
*Calls **stop()** (p. 485) then **clear()** (p. 483).*
- virtual void **execute** (const **Pointer**< **MessageDispatch** > &data)
Executes the dispatch.
- virtual void **executeFirst** (const **Pointer**< **MessageDispatch** > &data)
Executes the dispatch.
- virtual void **clearMessagesInProgress** ()
Removes all messages in the Dispatch Channel so that non are delivered.
- virtual bool **hasUnconsumedMessages** () const
- virtual void **wakeup** ()
wakeup this executor and dispatch any pending messages.
- virtual void **start** ()
Starts the dispatching.
- virtual void **stop** ()
Stops dispatching.

- virtual void **close** ()
Terminates the dispatching thread.
- virtual bool **isRunning** () const
- virtual bool **isEmpty** ()
- virtual void **clear** ()
Removes all queued messages and destroys them.
- virtual bool **iterate** ()
*Iterates on the **MessageDispatchChannel** (p. 2431) sending all pending messages to the Consumers they are destined for.*
- std::vector< **Pointer**< **MessageDispatch** > > **getUnconsumedMessages** ()

6.70.1 Detailed Description

Delegate dispatcher for a single session. Contains a thread to provide for asynchronous dispatching.

6.70.2 Constructor & Destructor Documentation

6.70.2.1 **activemq::core::ActiveMQSessionExecutor::ActiveMQSessionExecutor** (**ActiveMQSession** * *session*)

Creates an un-started executor for the given session.

6.70.2.2 **virtual** **activemq::core::ActiveMQSessionExecutor::~~ActiveMQSessionExecutor** () [virtual]

Calls **stop**() (p. 485) then **clear**() (p. 483).

6.70.3 Member Function Documentation

6.70.3.1 **virtual void activemq::core::ActiveMQSessionExecutor::clear** () [inline, virtual]

Removes all queued messages and destroys them.

6.70.3.2 **virtual void ac-** **tivemq::core::ActiveMQSessionExecutor::clearMessagesInProgress** () [inline, virtual]

Removes all messages in the Dispatch Channel so that non are delivered.

6.70.3.3 `virtual void activemq::core::ActiveMQSessionExecutor::close ()`
[inline, virtual]

Terminates the dispatching thread.

Once this is called, the executor is no longer usable.

6.70.3.4 `virtual void activemq::core::ActiveMQSessionExecutor::execute (const
Pointer< MessageDispatch > & data)` [virtual]

Executes the dispatch.

Adds the given data to the end of the queue.

Parameters

data - the data to be dispatched.

6.70.3.5 `virtual void activemq::core::ActiveMQSessionExecutor::executeFirst (`
`const Pointer< MessageDispatch > & data)` [virtual]

Executes the dispatch.

Adds the given data to the beginning of the queue.

Parameters

data - the data to be dispatched.

6.70.3.6 `std::vector< Pointer<MessageDispatch> > ac-`
`tivemq::core::ActiveMQSessionExecutor::getUnconsumedMessages ()`
[inline]

Returns

a vector containing all the unconsumed messages, this clears the Message Dispatch Channel when called.

6.70.3.7 `virtual bool ac-`
`tivemq::core::ActiveMQSessionExecutor::hasUncomsumedMessages ()`
`const` [inline, virtual]

Returns

true if there are any pending messages in the dispatch channel.

6.70.3.8 `virtual bool activemq::core::ActiveMQSessionExecutor::isEmpty ()`
[inline, virtual]

Returns

true if there are no messages in the Dispatch Channel.

6.70.3.9 `virtual bool activemq::core::ActiveMQSessionExecutor::isRunning ()`
`const [inline, virtual]`

Returns

true indicates if the executor is started

6.70.3.10 `virtual bool activemq::core::ActiveMQSessionExecutor::iterate ()`
`[virtual]`

Iterates on the **MessageDispatchChannel** (p. 2431) sending all pending messages to the Consumers they are destined for.

Returns

false if there are no more messages to dispatch.

Implements **activemq::threads::Task** (p. 3495).

6.70.3.11 `virtual void activemq::core::ActiveMQSessionExecutor::start ()`
`[virtual]`

Starts the dispatching.

6.70.3.12 `virtual void activemq::core::ActiveMQSessionExecutor::stop ()`
`[virtual]`

Stops dispatching.

6.70.3.13 `virtual void activemq::core::ActiveMQSessionExecutor::wakeup ()`
`[virtual]`

wakeup this executor and dispatch any pending messages.

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQSessionExecutor.h`

6.71 activemq::commands::ActiveMQStreamMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQStreamMessage.h>
```

Inheritance diagram for **activemq::commands::ActiveMQStreamMessage**:

Public Member Functions

- **ActiveMQStreamMessage** ()
- virtual **~ActiveMQStreamMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQStreamMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual void **onSend** ()
Store the Data that was written to the stream before a send.
- virtual **cms::StreamMessage * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual void **clearBody** () throw (cms::CMSException)
Clears out the body of the message.
- virtual void **reset** () throw (cms::CMSException)
Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)
Reads a Boolean from the Stream message stream.
- virtual void **writeBoolean** (bool value) throw (cms::MessageNotWriteableException, cms::CMSException)
Writes a boolean to the Stream message stream as a 1-byte value.
- virtual unsigned char **readByte** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)
Reads a Byte from the Stream message stream.
- virtual void **writeByte** (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSException)
Writes a byte to the Stream message stream as a 1-byte value.

- virtual int **readBytes** (std::vector< unsigned char > &value) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a byte array from the Stream message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.
- virtual int **readBytes** (unsigned char *buffer, int length) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a portion of the Stream message stream.
- virtual void **writeBytes** (const unsigned char *value, int offset, int length) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a portion of a byte array to the Stream message stream.
- virtual char **readChar** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Char from the Stream message stream.
- virtual void **writeChar** (char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a char to the Stream message stream as a 1-byte value.
- virtual float **readFloat** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 32 bit float from the Stream message stream.
- virtual void **writeFloat** (float value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a float to the Stream message stream as a 4 byte value.
- virtual double **readDouble** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 64 bit double from the Stream message stream.
- virtual void **writeDouble** (double value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a double to the Stream message stream as a 8 byte value.
- virtual short **readShort** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 16 bit signed short from the Stream message stream.

- virtual void **writeShort** (short value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed short to the Stream message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit unsigned short from the Stream message stream.

- virtual void **writeUnsignedShort** (unsigned short value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a unsigned short to the Stream message stream as a 2 byte value.

- virtual int **readInt** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit signed integer from the Stream message stream.

- virtual void **writeInt** (int value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed int to the Stream message stream as a 4 byte value.

- virtual long long **readLong** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit long from the Stream message stream.

- virtual void **writeLong** (long long value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a long long to the Stream message stream as a 8 byte value.

- virtual std::string **readString** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads an ASCII String from the Stream message stream.

- virtual void **writeString** (const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes an ASCII String to the Stream message stream.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQSTREAMMESSAGE** = 27

6.71.1 Constructor & Destructor Documentation

6.71.1.1 `activemq::commands::ActiveMQStreamMessage::ActiveMQStreamMessage ()`

6.71.1.2 `virtual
activemq::commands::ActiveMQStreamMessage::~~ActiveMQStreamMessage
() [virtual]`

6.71.2 Member Function Documentation

6.71.2.1 `virtual void activemq::commands::ActiveMQStreamMessage::clearBody ()
throw (cms::CMSException) [virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 383).

6.71.2.2 `virtual cms::StreamMessage* ac-
tivemq::commands::ActiveMQStreamMessage::clone ()
const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.71.2.3 `virtual ActiveMQStreamMessage* ac-
tivemq::commands::ActiveMQStreamMessage::cloneDataStructure ()
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2363).

6.71.2.4 `virtual void ac-
tivemq::commands::ActiveMQStreamMessage::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2363).

6.71.2.5 `virtual bool activemq::commands::ActiveMQStreamMessage::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 383).

6.71.2.6 `virtual unsigned char activemq::commands::ActiveMQStreamMessage::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Reimplemented from `activemq::commands::Message` (p. 2365).

6.71.2.7 `virtual void activemq::commands::ActiveMQStreamMessage::onSend ()` [virtual]

Store the Data that was written to the stream before a send.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 390).

6.71.2.8 `virtual bool activemq::commands::ActiveMQStreamMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)` [virtual]

Reads a Boolean from the Stream message stream.

Returns

boolean value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.9 virtual unsigned char activemq::commands::ActiveMQStreamMessage::readByte
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]

Reads a Byte from the Stream message stream.

Returns

unsigned char value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.10 virtual int activemq::commands::ActiveMQStreamMessage::readBytes
(unsigned char * *buffer*, int *length*) const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a portion of the Stream message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an CMSException is thrown. No bytes will be read from the stream for this exception case.

Parameters

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.11 `virtual int activemq::commands::ActiveMQStreamMessage::readBytes
(std::vector< unsigned char > & value) const throw (cms::MessageEOFException, cms::MessageFormatException,
cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a byte array from the Stream message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

value buffer to place data in

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.12 `virtual char activemq::commands::ActiveMQStreamMessage::readChar
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]`

Reads a Char from the Stream message stream.

Returns

char value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.13 virtual double activemq::commands::ActiveMQStreamMessage::readDouble
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]

Reads a 64 bit double from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.14 virtual float activemq::commands::ActiveMQStreamMessage::readFloat
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]

Reads a 32 bit float from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.15 virtual int activemq::commands::ActiveMQStreamMessage::readInt
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]

Reads a 32 bit signed integer from the Stream message stream.

Returns

int value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.16 virtual long long activemq::commands::ActiveMQStreamMessage::readLong
 () const throw (cms::MessageEOFException,
 cms::MessageFormatException, cms::MessageNotReadableException,
 cms::CMSException) [virtual]

Reads a 64 bit long from the Stream message stream.

Returns

long long value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.17 virtual short activemq::commands::ActiveMQStreamMessage::readShort
 () const throw (cms::MessageEOFException,
 cms::MessageFormatException, cms::MessageNotReadableException,
 cms::CMSException) [virtual]

Reads a 16 bit signed short from the Stream message stream.

Returns

short value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.18 virtual std::string activemq::commands::ActiveMQStreamMessage::readString
 () const throw (cms::MessageEOFException,
 cms::MessageFormatException, cms::MessageNotReadableException,
 cms::CMSException) [virtual]

Reads an ASCII String from the Stream message stream.

Returns

String from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.19 virtual unsigned short activemq::commands::ActiveMQStreamMessage::readUnsignedShort () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 16 bit unsigned short from the Stream message stream.

Returns

unsigned short value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.71.2.20 virtual void activemq::commands::ActiveMQStreamMessage::reset () throw (cms::CMSException) [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions

CMSException

6.71.2.21 virtual std::string activemq::commands::ActiveMQStreamMessage::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p.2372).

6.71.2.22 `virtual void activemq::commands::ActiveMQStreamMessage::writeBoolean (bool value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a boolean to the Stream message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

value boolean to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.71.2.23 `virtual void activemq::commands::ActiveMQStreamMessage::writeByte (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a byte to the Stream message stream as a 1-byte value.

Parameters

value byte to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.71.2.24 `virtual void activemq::commands::ActiveMQStreamMessage::writeBytes (const unsigned char * value, int offset, int length) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a portion of a byte array to the Stream message stream.

size as the number of bytes to write.

Parameters

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.71.2.25 virtual void activemq::commands::ActiveMQStreamMessage::writeBytes
(const std::vector< unsigned char > & *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

Parameters

value bytes to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.71.2.26 virtual void activemq::commands::ActiveMQStreamMessage::writeChar
(char *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a char to the Stream message stream as a 1-byte value.

Parameters

value char to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.71.2.27 virtual void activemq::commands::ActiveMQStreamMessage::writeDouble
(double *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a double to the Stream message stream as a 8 byte value.

Parameters

value double to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.71.2.28 virtual void activemq::commands::ActiveMQStreamMessage::writeFloat
(float *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a float to the Stream message stream as a 4 byte value.

Parameters

value float to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.71.2.29 virtual void activemq::commands::ActiveMQStreamMessage::writeInt
(int *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a signed int to the Stream message stream as a 4 byte value.

Parameters

value signed int to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.71.2.30 virtual void activemq::commands::ActiveMQStreamMessage::writeLong
(long long *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a long long to the Stream message stream as a 8 byte value.

Parameters

value signed long long to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.71.2.31 virtual void activemq::commands::ActiveMQStreamMessage::writeShort
(short *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a signed short to the Stream message stream as a 2 byte value.

Parameters

value signed short to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.72

activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
Class Reference

501

```
6.71.2.32 virtual void activemq::commands::ActiveMQStreamMessage::writeString  
        ( const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSException  
        ) [virtual]
```

Writes an ASCII String to the Stream message stream.

Parameters

value String to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

```
6.71.2.33 virtual void ac-  
        tivemq::commands::ActiveMQStreamMessage::writeUnsignedShort ( unsigned short value ) throw ( cms::MessageNotWriteableException,  
        cms::CMSException ) [virtual]
```

Writes a unsigned short to the Stream message stream as a 2 byte value.

Parameters

value unsigned short to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.71.3 Field Documentation

```
6.71.3.1 const unsigned char activemq::commands::ActiveMQStreamMessage::ID_ -  
        ACTIVEMQSTREAMMESSAGE = 27 [static]
```

The documentation for this class was generated from the following file:

- src/main/activemq/commands/ActiveMQStreamMessage.h

6.72 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 499).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.72.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 499).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.72.2 Constructor & Destructor Documentation

6.72.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller () [inline]`

6.72.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller () [inline, virtual]`

6.72.3 Member Function Documentation

6.72.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.72.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.72.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

6.72.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529).

6.72.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2530).

6.72.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

6.73

activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller

Class Reference

505

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2531).

```
6.72.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightUnm
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2531).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h

6.73 activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 503).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.73.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 503).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.73.2 Constructor & Destructor Documentation

6.73.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller () [inline]`

6.73.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller () [inline, virtual]`

6.73.3 Member Function Documentation

6.73.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.73.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.73.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

6.73.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

6.73.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2542).

6.73.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

6.74

activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller

Class Reference

509

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2543).

```
6.73.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightUnm
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2543).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h

6.74 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 507).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.74.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 507).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.74.2 Constructor & Destructor Documentation

6.74.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller () [inline]`

6.74.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller () [inline, virtual]`

6.74.3 Member Function Documentation

6.74.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.74.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.74.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

6.74.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537).

6.74.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2538).

6.74.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

6.75

activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

Class Reference

513

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2539).

```
6.74.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightUnm
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2539).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h

6.75 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 511).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller**:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual ~**ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.75.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 511).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.75.2 Constructor & Destructor Documentation

6.75.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller () [inline]`

6.75.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller () [inline, virtual]`

6.75.3 Member Function Documentation

6.75.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.75.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.75.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2524).

6.75.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2525).

6.75.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

6.75.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

6.76

activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller

Class Reference

517

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2526).

```
6.75.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightUnm
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2527).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h

6.76 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 515).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.76.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 515).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.76.2 Constructor & Destructor Documentation

6.76.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller () [inline]`

6.76.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller () [inline, virtual]`

6.76.3 Member Function Documentation

6.76.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.76.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.76.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

6.76.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

6.76.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2534).

6.76.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

6.77

activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller

Class Reference

521

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2535).

6.76.3.7 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightUnm  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataInputStream * dataIn,  
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2535).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQStreamMessageMarshaller.h**

6.77 activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 519).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller**:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.77.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 519).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.77.2 Constructor & Destructor Documentation

6.77.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller () [inline]`

6.77.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller () [inline, virtual]`

6.77.3 Member Function Documentation

6.77.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.77.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.77.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

6.77.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

6.77.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2546).

6.77.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2547).

```
6.77.3.7 virtual void ac-
      tivemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightUnm
      ( OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataInputStream * dataIn,
      utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2547).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamMessageMarshaller.h`

6.78 activemq::commands::ActiveMQTempDestination Class Reference

```
#include <src/main/activemq/commands/ActiveMQTempDestination.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTempDestination`:

Public Member Functions

- **ActiveMQTempDestination** ()
- **ActiveMQTempDestination** (const std::string &name)
- virtual ~**ActiveMQTempDestination** ()
- virtual unsigned char **getDataStructureType** () const
- virtual **ActiveMQTempDestination** * **cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

- virtual void **close** () throw (cms::CMSException)

Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker.

- void **setConnection** (core::ActiveMQConnection *connection)

Sets the Parent Connection that is notified when this destination is destroyed.

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQTEMPDESTINATION** = 0

Protected Attributes

- core::ActiveMQConnection * **connection**

Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

6.78.1 Constructor & Destructor Documentation

- 6.78.1.1** `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination ()`
- 6.78.1.2** `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination (const std::string & name)`
- 6.78.1.3** `virtual
activemq::commands::ActiveMQTempDestination::~~ActiveMQTempDestination () [virtual]`

6.78.2 Member Function Documentation

- 6.78.2.1** `virtual ActiveMQTempDestination* activemq::commands::ActiveMQTempDestination::cloneDataStructure () const [inline, virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

- 6.78.2.2** `virtual void activemq::commands::ActiveMQTempDestination::close () throw (cms::CMSException) [virtual]`

Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker.

This should only be called when the user is certain that they are finished with this destination. The TempDestination is not closed automatically on shutdown. throws **cms::CMSException** (p.1074)

Implements **cms::Closeable** (p.1065).

- 6.78.2.3** `virtual void activemq::commands::ActiveMQTempDestination::copyDataStructure (const DataStructure * src) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

References `activemq::commands::ActiveMQDestination::copyDataStructure()`.

- 6.78.2.4** `virtual bool activemq::commands::ActiveMQTempDestination::equals (const DataStructure * value) const [inline, virtual]`

Compares the **DataStructure** (p.1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

References `activemq::commands::ActiveMQTempDestination::equals()`.

6.78.2.5 `virtual unsigned char activemq::commands::ActiveMQTempDestination::getDataStructureType () const [virtual]`

6.78.2.6 `void activemq::commands::ActiveMQTempDestination::setConnection (core::ActiveMQConnection * connection) [inline]`

Sets the Parent Connection that is notified when this destination is destroyed.

Parameters

connection - The parent connection.

6.78.2.7 `virtual std::string activemq::commands::ActiveMQTempDestination::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.78.3 Field Documentation

6.78.3.1 `core::ActiveMQConnection* activemq::commands::ActiveMQTempDestination::connection [protected]`

Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

6.78.3.2 `const unsigned char activemq::commands::ActiveMQTempDestination::ID_ - ACTIVEMQTEMPDESTINATION = 0 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempDestination.h`

Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 527).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.79.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 527).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.79.2 Constructor & Destructor Documentation

6.79.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` [inline]

6.79.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()` [inline, virtual]

6.79.3 Member Function Documentation

6.79.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 291).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 554), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 582).

6.79.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn)` throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

6.79

activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller

Class Reference

531

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 291).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 555), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 583).

```
6.79.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMa
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 292).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 555), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 583).

```
6.79.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMa
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 292).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 555), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 583).

```
6.79.3.5 virtual void ac-
      tivemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightUn
      ( OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataInputStream * dataIn,
      utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 293).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 556), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 584).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h`

6.80 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDesti` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempDestinationMarshaller` (p. 530).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller`:

Public Member Functions

- `ActiveMQTempDestinationMarshaller ()`

6.80

activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller
Class Reference 533

-
- virtual `~ActiveMQTempDestinationMarshaller()`
 - virtual void **tightUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Un-marshal an object instance from the data input stream.
 - virtual int **tightMarshal1** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Write the booleans that this object uses to a BooleanStream.
 - virtual void **tightMarshal2** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Write a object instance to data output stream.
 - virtual void **looseUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`) throw (`decaf::io::IOException`)
Un-marshal an object instance from the data input stream.
 - virtual void **looseMarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`) throw (`decaf::io::IOException`)
Write a object instance to data output stream.

6.80.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 530).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.80.2 Constructor & Destructor Documentation

- 6.80.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()** [inline]
- 6.80.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()** [inline, virtual]

6.80.3 Member Function Documentation

- 6.80.3.1 **virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::looseMarshal1** (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 295).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 558), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 590).

6.80.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 295).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 559), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 591).

6.80.3.3 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled

6.80

activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller
Class Reference 535

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 296).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 559), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 591).

6.80.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMa
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 296).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 559), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 591).

6.80.3.5 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightUn
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 297).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 560), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 592).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h`

6.81 activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 534).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.81

activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
Class Reference 537

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.81.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 534).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.81.2 Constructor & Destructor Documentation

6.81.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller** () [inline]

6.81.2.2 **virtual**
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller () [inline, virtual]

6.81.3 Member Function Documentation

6.81.3.1 **virtual void** **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::looseMarshal** (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 299).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 562), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 586).

```

6.81.3.2 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::looseUn-
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 299).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 563), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 587).

```

6.81.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMa-
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 300).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 563), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 587).


```

6.81.3.4 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMa
        ( OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 300).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 563), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 587).

```

6.81.3.5 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightUnm
        ( OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataInputStream * dataIn,
        utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 301).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 564), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 588).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempDestinationMarshaller.h**

6.82 activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 538).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.82.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 538).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.82.2 Constructor & Destructor Documentation

6.82.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` [inline]

6.82.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()` [inline, virtual]

6.82.3 Member Function Documentation

6.82.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 303).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 566), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 594).

6.82.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn)` throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 303).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 567), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 595).

```
6.82.3.3 virtual int ac-
          tivemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMa
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
          ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 304).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 567), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 595).

```
6.82.3.4 virtual void ac-
          tivemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMa
          ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

6.83

activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller

Class Reference

543

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 304).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 567), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 595).

6.82.3.5 virtual void ac-

```
timemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightUn  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataInputStream * dataIn,  
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 305).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 568), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 596).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h`

6.83 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestin

Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 541).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller**:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()

- virtual `~ActiveMQTempDestinationMarshaller()`
- virtual void **tightUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`) throw (`decaf::io::IOException`)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`) throw (`decaf::io::IOException`)
Write a object instance to data output stream.

6.83.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 541).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.83.2 Constructor & Destructor Documentation

- 6.83.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` [inline]
- 6.83.2.2 virtual `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()` [inline, virtual]

6.83.3 Member Function Documentation

- 6.83.3.1 virtual void `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::looseMarshal1` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 307).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 570), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 598).

6.83.3.2 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::looseUn-
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 307).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 571), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 599).

6.83.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMa
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 308).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 571), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 599).

```
6.83.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMa
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 308).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 571), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 599).

```
6.83.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightUn
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 309).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 572), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 600).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h`

6.84 **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 545).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller**:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.84.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 545).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.84.2 Constructor & Destructor Documentation

6.84.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller** () [inline]

6.84.2.2 **virtual**
activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller () [inline, virtual]

6.84.3 Member Function Documentation

6.84.3.1 **virtual void** **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::looseMarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 311).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 574), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 602).

```

6.84.3.2 virtual void ac-
          tivemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::looseUn-
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, decaf::io::DataInputStream * dataIn ) throw (
            decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 311).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 575), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 603).

```

6.84.3.3 virtual int ac-
          tivemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightMa-
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
            ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 312).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 575), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 603).

```

6.84.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightMa
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 312).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 575), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 603).

```

6.84.3.5 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightUnm
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 313).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 576), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 604).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h`

6.85 activemq::commands::ActiveMQTempQueue Class Reference

#include <src/main/activemq/commands/ActiveMQTempQueue.h>

Inheritance diagram for activemq::commands::ActiveMQTempQueue:

Public Member Functions

- **ActiveMQTempQueue** ()
- **ActiveMQTempQueue** (const std::string &name)
- virtual ~**ActiveMQTempQueue** ()
- virtual unsigned char **getDataStructureType** () const
- virtual **ActiveMQTempQueue** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
Converts the Destination Name into a String.
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const
Retrieve the Destination Type for this Destination.
- virtual **cms::Destination** * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)
Copies the contents of the given Destinastion object to this one.
- virtual const **cms::CMSProperties** & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual std::string **getQueueName** () const throw (cms::CMSException)
Gets the name of this queue.
- virtual void **destroy** () throw (cms::CMSException)
Destroy's the Temp Destination at the Broker.

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQTEMPQUEUE** = 102

6.85.1 Constructor & Destructor Documentation

6.85.1.1 `activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue ()`

6.85.1.2 `activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue (const std::string & name)`

6.85.1.3 `virtual
activemq::commands::ActiveMQTempQueue::~~ActiveMQTempQueue ()` [virtual]

6.85.2 Member Function Documentation

6.85.2.1 `virtual cms::Destination* activemq::commands::ActiveMQTempQueue::clone ()`
`const` [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p.1612).

6.85.2.2 `virtual ActiveMQTempQueue* activemq::commands::ActiveMQTempQueue::cloneDataStructure () const`
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.85.2.3 `virtual void activemq::commands::ActiveMQTempQueue::copy (const cms::Destination & source)` [inline, virtual]

Copies the contents of the given Destination object to this one.

Parameters

source The source Destination object.

Implements **cms::Destination** (p.1612).

6.85.2.4 `virtual void activemq::commands::ActiveMQTempQueue::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

6.85.2.5 `virtual void activemq::commands::ActiveMQTempQueue::destroy ()
throw (cms::CMSException) [virtual]`

Destroy's the Temp Destination at the Broker.

Exceptions

CMSException

Implements **cms::TemporaryQueue** (p. 3517).

6.85.2.6 `virtual bool activemq::commands::ActiveMQTempQueue::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.85.2.7 `virtual const cms::Destination* ac-
tivemq::commands::ActiveMQTempQueue::getCMSDestination () const
[inline, virtual]`

Returns

the **cms::Destination** (p. 1610) interface pointer that the objects that derive from this class implement.

6.85.2.8 `virtual const cms::CMSProperties& ac-
tivemq::commands::ActiveMQTempQueue::getCMSProperties () const
[inline, virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

const reference to a properties object.

Implements **cms::Destination** (p. 1612).

6.85.2.9 `virtual unsigned char activemq::commands::ActiveMQTempQueue::getDataStructureType () const [virtual]`

6.85.2.10 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempQueue::getDestinationType () const [inline, virtual]`

Retrieve the Destination Type for this Destination.

Returns

The Destination Type

Implements `cms::Destination` (p. 1612).

6.85.2.11 `virtual std::string activemq::commands::ActiveMQTempQueue::getQueueName () const throw (cms::CMSException) [inline, virtual]`

Gets the name of this queue.

Returns

The queue name.

Implements `cms::TemporaryQueue` (p. 3517).

6.85.2.12 `virtual std::string activemq::commands::ActiveMQTempQueue::toString () const [virtual]`

Converts the Destination Name into a String.

Returns

string name

6.85.3 Field Documentation

6.85.3.1 `const unsigned char activemq::commands::ActiveMQTempQueue::ID_-ACTIVEMQTEMPQUEUE = 102 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempQueue.h`

6.86 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueue Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempQueueMarshaller` (p. 552).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.86.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 552).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.86.2 Constructor & Destructor Documentation

6.86.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller () [inline]`

6.86.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller () [inline, virtual]`

6.86.3 Member Function Documentation

6.86.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.86.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.86.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 528).

6.86.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 528).

6.86.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 529).

6.86.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 529).

```
6.86.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 530).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h`

6.87 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempQueueMarshaller` (p. 556).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller`:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.87.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p.556).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.87.2 Constructor & Destructor Documentation

6.87.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller () [inline]`

6.87.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller () [inline, virtual]`

6.87.3 Member Function Documentation

6.87.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.87.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.87.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 531).

6.87.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 532).

6.87.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 532).

6.87.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 533).

```
6.87.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 533).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h`

6.88 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempQueueMarshaller` (p. 560).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.88.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 560).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.88.2 Constructor & Destructor Documentation

6.88.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller () [inline]`

6.88.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller () [inline, virtual]`

6.88.3 Member Function Documentation

6.88.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.88.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.88.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 535).

6.88.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 536).

6.88.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 536).

6.88.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 537).

```
6.88.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 537).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h`

6.89 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempQueueMarshaller` (p. 564).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.89.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 564).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.89.2 Constructor & Destructor Documentation

6.89.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller () [inline]`

6.89.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller () [inline, virtual]`

6.89.3 Member Function Documentation

6.89.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.89.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.89.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 539).

6.89.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 539).

6.89.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 540).

6.89.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 540).

```
6.89.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 541).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h`

6.90 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempQueueMarshaller` (p. 568).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller`:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.90.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 568).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.90.2 Constructor & Destructor Documentation

6.90.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller () [inline]`

6.90.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller () [inline, virtual]`

6.90.3 Member Function Documentation

6.90.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.90.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.90.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 542).

6.90.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 543).

6.90.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 543).

6.90.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 544).

```
6.90.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 544).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h`

6.91 `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempQueueMarshaller` (p. 572).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller`:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.91.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 572).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.91.2 Constructor & Destructor Documentation

6.91.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller () [inline]`

6.91.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller () [inline, virtual]`

6.91.3 Member Function Documentation

6.91.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.91.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.91.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 546).

6.91.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 547).

6.91.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 547).

6.91.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 548).

```
6.91.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 548).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarshaller.h`

6.92 `activemq::commands::ActiveMQTempTopic` Class Reference

```
#include <src/main/activemq/commands/ActiveMQTempTopic.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTempTopic`:

Public Member Functions

- **ActiveMQTempTopic** ()
- **ActiveMQTempTopic** (const std::string &name)
- virtual ~**ActiveMQTempTopic** ()
- virtual unsigned char **getDataStructureType** () const
- virtual **ActiveMQTempTopic** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
Converts the Destination Name into a String.
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const
Retrieve the Destination Type for this Destination.
- virtual **cms::Destination** * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)
Copies the contents of the given Destinastion object to this one.
- virtual const **cms::CMSProperties** & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual std::string **getTopicName** () const throw (cms::CMSException)
Gets the name of this topic.
- virtual void **destroy** () throw (cms::CMSException)
Destroy's the Temp Destination at the Broker.

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQTEMPTOPIC** = 103

6.92.1 Constructor & Destructor Documentation

6.92.1.1 `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic ()`

6.92.1.2 `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic (const std::string & name)`

6.92.1.3 `virtual
activemq::commands::ActiveMQTempTopic::~~ActiveMQTempTopic ()
[virtual]`

6.92.2 Member Function Documentation

6.92.2.1 `virtual cms::Destination* ac-
tivemq::commands::ActiveMQTempTopic::clone ()
const [inline, virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p.1612).

6.92.2.2 `virtual ActiveMQTempTopic* ac-
tivemq::commands::ActiveMQTempTopic::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.92.2.3 `virtual void activemq::commands::ActiveMQTempTopic::copy (const cms::Destination & source) [inline, virtual]`

Copies the contents of the given Destination object to this one.

Parameters

source The source Destination object.

Implements **cms::Destination** (p.1612).

6.92.2.4 `virtual void ac-
tivemq::commands::ActiveMQTempTopic::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

6.92.2.5 `virtual void activemq::commands::ActiveMQTempTopic::destroy ()
throw (cms::CMSException) [virtual]`

Destroy's the Temp Destination at the Broker.

Exceptions

CMSException

Implements **cms::TemporaryTopic** (p. 3518).

6.92.2.6 `virtual bool activemq::commands::ActiveMQTempTopic::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.92.2.7 `virtual const cms::Destination* ac-
tivemq::commands::ActiveMQTempTopic::getCMSDestination () const
[inline, virtual]`

Returns

the **cms::Destination** (p. 1610) interface pointer that the objects that derive from this class implement.

6.92.2.8 `virtual const cms::CMSPProperties& ac-
tivemq::commands::ActiveMQTempTopic::getCMSPProperties () const
[inline, virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

const reference to a properties object.

Implements **cms::Destination** (p. 1612).

6.92.2.9 `virtual unsigned char activemq::commands::ActiveMQTempTopic::getDataStructureType () const [virtual]`

6.92.2.10 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempTopic::getDestinationType () const [inline, virtual]`

Retrieve the Destination Type for this Destination.

Returns

The Destination Type

Implements **cms::Destination** (p. 1612).

6.92.2.11 `virtual std::string activemq::commands::ActiveMQTempTopic::getTopicName () const throw (cms::CMSException) [inline, virtual]`

Gets the name of this topic.

Returns

The topic name.

Implements **cms::TemporaryTopic** (p. 3518).

6.92.2.12 `virtual std::string activemq::commands::ActiveMQTempTopic::toString () const [virtual]`

Converts the Destination Name into a String.

Returns

string name

6.92.3 Field Documentation

6.92.3.1 `const unsigned char activemq::commands::ActiveMQTempTopic::ID _ - ACTIVEMQTEMPTOPIC = 103 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempTopic.h`

6.93 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopic Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 580).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.93.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.580). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.93.2 Constructor & Destructor Documentation

6.93.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller () [inline]`

6.93.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller () [inline, virtual]`

6.93.3 Member Function Documentation

6.93.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.93.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.93.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 528).

6.93.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 528).

6.93.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal1(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 529).

6.93.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 529).

```
6.93.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 530).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h`

6.94 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempTopicMarshaller` (p. 584).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.94.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.584). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.94.2 Constructor & Destructor Documentation

6.94.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller () [inline]`

6.94.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller () [inline, virtual]`

6.94.3 Member Function Documentation

6.94.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.94.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.94.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 535).

6.94.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 536).

6.94.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal1(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 536).

6.94.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 537).

```
6.94.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 537).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h`

6.95 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempTopicMarshaller` (p. 588).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller`:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.95.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.588). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.95.2 Constructor & Destructor Documentation

6.95.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller () [inline]`

6.95.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller () [inline, virtual]`

6.95.3 Member Function Documentation

6.95.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.95.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.95.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 531).

6.95.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 532).

6.95.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 532).

6.95.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 533).

```
6.95.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 533).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h`

6.96 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempTopicMarshaller` (p. 592).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.96.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.592). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.96.2 Constructor & Destructor Documentation

6.96.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller () [inline]`

6.96.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller () [inline, virtual]`

6.96.3 Member Function Documentation

6.96.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.96.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.96.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 539).

6.96.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 539).

6.96.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal1(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 540).

6.96.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 540).

```
6.96.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 541).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h`

6.97 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempTopicMarshaller` (p. 596).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller`:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.97.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.596). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.97.2 Constructor & Destructor Documentation

6.97.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller () [inline]`

6.97.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller () [inline, virtual]`

6.97.3 Member Function Documentation

6.97.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.97.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.97.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 542).

6.97.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 543).

6.97.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 543).

6.97.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 544).

```
6.97.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 544).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h`

6.98 `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempTopicMarshaller` (p. 600).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller`:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.98.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.600). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.98.2 Constructor & Destructor Documentation

6.98.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller () [inline]`

6.98.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller () [inline, virtual]`

6.98.3 Member Function Documentation

6.98.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.98.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.98.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 546).

6.98.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 547).

6.98.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 547).

6.98.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 548).

```
6.98.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 548).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h`

6.99 `activemq::commands::ActiveMQTextMessage` Class Reference

```
#include <src/main/activemq/commands/ActiveMQTextMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTextMessage`:

Public Member Functions

- **ActiveMQTextMessage** ()
- virtual **~ActiveMQTextMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQTextMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw (cms::CMSException)
Clears out the body of the message.
- virtual void **beforeMarshal** (wireformat::WireFormat *wireFormat) throw (decaf::io::IOException)
*Performs any cleanup or other tasks that must be done before the **Message** (p. 2358) is marshalled to its binary WireFormat version.*
- virtual unsigned int **getSize** () const
Returns the Size of this message in Bytes.
- virtual **cms::TextMessage * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual std::string **getText** () const throw (cms::CMSException)
Gets the message character buffer.
- virtual void **setText** (const char *msg) throw (cms::MessageNotWriteableException, cms::CMSException)
Sets the message contents, does not take ownership of the passed char, but copies it instead.*
- virtual void **setText** (const std::string &msg) throw (cms::MessageNotWriteableException, cms::CMSException)
Sets the message contents.

Data Fields

- std::auto_ptr< std::string > **text**

Static Public Attributes

- static const unsigned char `ID_ACTIVEMQTEXTMESSAGE` = 28

6.99.1 Constructor & Destructor Documentation

6.99.1.1 `activemq::commands::ActiveMQTextMessage::ActiveMQTextMessage ()`

6.99.1.2 `virtual
activemq::commands::ActiveMQTextMessage::~~ActiveMQTextMessage () [virtual]`

6.99.2 Member Function Documentation

6.99.2.1 `virtual void activemq::commands::ActiveMQTextMessage::beforeMarshal
(wireformat::WireFormat * wireFormat) throw (decaf::io::IOException
) [virtual]`

Performs any cleanup or other tasks that must be done before the **Message** (p. 2358) is marshalled to its binary WireFormat version.

Parameters

wireFormat the WireFormat instance that is marshalling this message.

Implements `activemq::wireformat::MarshalAware` (p. 2329).

6.99.2.2 `virtual void activemq::commands::ActiveMQTextMessage::clearBody ()
throw (cms::CMSException) [virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 383).

6.99.2.3 `virtual cms::TextMessage* ac-
tivemq::commands::ActiveMQTextMessage::clone ()
const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.99.2.4 `virtual ActiveMQTextMessage* activemq::commands::ActiveMQTextMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2363).

6.99.2.5 `virtual void activemq::commands::ActiveMQTextMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2363).

6.99.2.6 `virtual bool activemq::commands::ActiveMQTextMessage::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 383).

6.99.2.7 `virtual unsigned char activemq::commands::ActiveMQTextMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1553) type copy.

Reimplemented from `activemq::commands::Message` (p. 2365).

6.99.2.8 `virtual unsigned int activemq::commands::ActiveMQTextMessage::getSize () const [virtual]`

Returns the Size of this message in Bytes.

Returns

number of bytes this message equates to.

Reimplemented from `activemq::commands::Message` (p. 2367).

6.99.2.9 `virtual std::string activemq::commands::ActiveMQTextMessage::getText () const throw (cms::CMSEException) [virtual]`

Gets the message character buffer.

Returns

The message character buffer.

Exceptions

CMSEException - if an internal error occurs.

6.99.2.10 `virtual void activemq::commands::ActiveMQTextMessage::setText (const std::string & msg) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets the message contents.

Parameters

msg The message buffer.

Exceptions

CMSEException - if an internal error occurs.

MessageNotWriteableException - if the message is in read-only mode..

6.99.2.11 `virtual void activemq::commands::ActiveMQTextMessage::setText (const char * msg) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters

msg The message buffer.

Exceptions

CMSEException - if an internal error occurs.

MessageNotWriteableException - if the message is in read-only mode..

6.100

activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller

Class Reference

611

6.99.2.12 `virtual std::string activemq::commands::ActiveMQTextMessage::toString
() const [virtual]`

Returns a string containing the information for this **DataSet** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2372).

6.99.3 Field Documentation

6.99.3.1 `const unsigned char activemq::commands::ActiveMQTextMessage::ID_-
ACTIVEMQTEXTMESSAGE = 28 [static]`

6.99.3.2 `std::auto_ptr<std::string> ac-
tivemq::commands::ActiveMQTextMessage::text
[mutable]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTextMessage.h`

6.100 **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 609).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataSet * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataSetType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **com-
mands::DataSet** *dataStructure, **decaf::io::DataInputStream** *dataIn,
utils::BooleanStream *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.100.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 609).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.100.2 Constructor & Destructor Documentation

6.100.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller**
 () [inline]

6.100.2.2 **virtual**
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller
 () [inline, virtual]

6.100.3 Member Function Documentation

6.100.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::createObject**
 () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.100

activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller

Class Reference

613

6.100.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.100.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2529).

6.100.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2529).

6.100.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2530).

6.100.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2531).

6.100.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2531).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h`

6.101 activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQTextMessageMarshaller` (p. 613).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`:

Public Member Functions

- `ActiveMQTextMessageMarshaller ()`
- `virtual ~ActiveMQTextMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.101.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 613).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.101.2 Constructor & Destructor Documentation

- 6.101.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller** () [inline]
- 6.101.2.2 virtual **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller** () [inline, virtual]

6.101.3 Member Function Documentation

- 6.101.3.1 virtual **commands::DataStructure*** **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

- 6.101.3.2 virtual unsigned char **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

6.101

activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller

Class Reference

617

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.101.3.3 **virtual void ac-**
tivemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseMarsh
(**OpenWireFormat * wireFormat**, **commands::DataStructure * dataStructure**, **decaf::io::DataOutputStream * dataOut**) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2537).

6.101.3.4 **virtual void ac-**
tivemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseUnmar
(**OpenWireFormat * wireFormat**, **commands::DataStructure * dataStructure**, **decaf::io::DataInputStream * dataIn**) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2537).

6.101.3.5 **virtual int ac-**
tivemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarsh
(**OpenWireFormat * wireFormat**, **commands::DataStructure * dataStructure**, **utils::BooleanStream * bs**) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2538).

```
6.101.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539).

```
6.101.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightUnmar
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - `BooleanStream` stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h`

6.102 activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQTextMessageMarshaller` (p. 617).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`:

Public Member Functions

- `ActiveMQTextMessageMarshaller ()`
- `virtual ~ActiveMQTextMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.102.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 617).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.102.2 Constructor & Destructor Documentation

6.102.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller () [inline]

6.102.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller () [inline, virtual]

6.102.3 Member Function Documentation

6.102.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.102.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.102

activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller

Class Reference

621

```
6.102.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseMarsh
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

```
6.102.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseUnmar
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541).

```
6.102.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2542).

```
6.102.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543).

```
6.102.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightUnmar
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h`

6.103 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQTextMessageMarshaller` (p. 621).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`:

Public Member Functions

- `ActiveMQTextMessageMarshaller ()`
- `virtual ~ActiveMQTextMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.103.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 621).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.103.2 Constructor & Destructor Documentation

6.103.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller () [inline]`

6.103.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller () [inline, virtual]`

6.103.3 Member Function Documentation

6.103.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.103.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.103

activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller

Class Reference

625

```
6.103.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseMarsh
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller**
(p. 2524).

```
6.103.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseUnmar
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller**
(p. 2525).

```
6.103.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

```
6.103.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526).

```
6.103.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightUnmar
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2527).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h`

6.104 activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQTextMessageMarshaller` (p. 625).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller`:

Public Member Functions

- `ActiveMQTextMessageMarshaller ()`
- `virtual ~ActiveMQTextMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.104.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 625).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.104.2 Constructor & Destructor Documentation

6.104.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller () [inline]`

6.104.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller () [inline, virtual]`

6.104.3 Member Function Documentation

6.104.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.104.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```

6.104.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::looseMarsh
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

```

6.104.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::looseUnmar
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545).

```

6.104.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightMarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2546).

```
6.104.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightMarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2547).

```
6.104.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightUnmar
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2547).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMessageMarshaller.h`

6.105 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQTextMessageMarshaller` (p. 629).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`:

Public Member Functions

- `ActiveMQTextMessageMarshaller ()`
- `virtual ~ActiveMQTextMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.105.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 629).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.105.2 Constructor & Destructor Documentation

6.105.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller () [inline]

6.105.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller () [inline, virtual]

6.105.3 Member Function Documentation

6.105.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.105.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```

6.105.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseMarsh
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

```

6.105.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseUnmar
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2533).

```

6.105.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2534).

```
6.105.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535).

```
6.105.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightUnmar
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2535).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h`

6.106 activemq::commands::ActiveMQTopic Class Reference

```
#include <src/main/activemq/commands/ActiveMQTopic.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTopic`:

Public Member Functions

- **ActiveMQTopic** ()
- **ActiveMQTopic** (const std::string &name)
- virtual **~ActiveMQTopic** ()
- virtual unsigned char **getDataStructureType** () const
- virtual **ActiveMQTopic * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const
Retrieve the Destination Type for this Destination.
- virtual **cms::Destination** * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)
Copies the contents of the given Destination object to this one.

- virtual const **cms::CMSProperties** & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual std::string **getTopicName** () const throw (cms::CMSException)
Gets the name of this topic.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTOPIC** = 101

6.106.1 Constructor & Destructor Documentation

- 6.106.1.1** **activemq::commands::ActiveMQTopic::ActiveMQTopic** ()
- 6.106.1.2** **activemq::commands::ActiveMQTopic::ActiveMQTopic** (const
std::string & *name*)
- 6.106.1.3** **virtual activemq::commands::ActiveMQTopic::~~ActiveMQTopic** ()
[virtual]

6.106.2 Member Function Documentation

- 6.106.2.1** **virtual cms::Destination* activemq::commands::ActiveMQTopic::clone** () const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p.1612).

- 6.106.2.2** **virtual ActiveMQTopic* activemq::commands::ActiveMQTopic::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

- 6.106.2.3** **virtual void activemq::commands::ActiveMQTopic::copy** (const
cms::Destination & *source*) [inline, virtual]

Copies the contents of the given Destination object to this one.

Parameters

source The source Destination object.

Implements **cms::Destination** (p.1612).

6.106.2.4 `virtual void activemq::commands::ActiveMQTopic::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

6.106.2.5 `virtual bool activemq::commands::ActiveMQTopic::equals (const DataStructure * value) const [inline, virtual]`

Compares the **DataStructure** (p.1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

References `activemq::commands::ActiveMQDestination::equals()`.

6.106.2.6 `virtual const cms::Destination* activemq::commands::ActiveMQTopic::getCMSDestination () const [inline, virtual]`

Returns

the **cms::Destination** (p.1610) interface pointer that the objects that derive from this class implement.

6.106.2.7 `virtual const cms::CMSProperties& activemq::commands::ActiveMQTopic::getCMSProperties () const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

const reference to a properties object.

Implements **cms::Destination** (p.1612).

6.106.2.8 `virtual unsigned char activemq::commands::ActiveMQTopic::getDataStructureType () const [virtual]`

6.106.2.9 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTopic::getDestinationType () const [inline, virtual]`

Retrieve the Destination Type for this Destination.

Returns

The Destination Type

Implements **cms::Destination** (p. 1612).

6.106.2.10 `virtual std::string activemq::commands::ActiveMQTopic::getTopicName () const throw (cms::CMSException) [inline, virtual]`

Gets the name of this topic.

Returns

The topic name.

Implements **cms::Topic** (p. 3568).

6.106.2.11 `virtual std::string activemq::commands::ActiveMQTopic::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.106.3 Field Documentation

6.106.3.1 `const unsigned char activemq::commands::ActiveMQTopic::ID_ - ACTIVEMQTOPIC = 101 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTopic.h`

6.107 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 636).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.107.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p.636). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.107.2 Constructor & Destructor Documentation

6.107.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller () [inline]`

6.107.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller () [inline, virtual]`

6.107.3 Member Function Documentation

6.107.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.107.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.107.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 291).

6.107.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 291).

6.107.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 292).

6.107.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 292).

```
6.107.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 293).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h`

6.108 `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTopicMarshaller` (p. 640).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.108.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 640). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.108.2 Constructor & Destructor Documentation

6.108.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` [inline]

6.108.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` [inline, virtual]

6.108.3 Member Function Documentation

6.108.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.108.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.108.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 299).

6.108.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 299).

6.108.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 300).

6.108.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 300).

```
6.108.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 301).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h`

6.109 `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTopicMarshaller` (p. 644).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller`:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.109.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p.644). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.109.2 Constructor & Destructor Documentation

6.109.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller () [inline]`

6.109.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller () [inline, virtual]`

6.109.3 Member Function Documentation

6.109.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.109.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.109.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 295).

6.109.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 295).

6.109.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 296).

6.109.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 296).

```
6.109.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 297).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h`

6.110 `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTopicMarshaller` (p. 648).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.110.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p.648). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.110.2 Constructor & Destructor Documentation

6.110.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller() [inline]`

6.110.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller() [inline, virtual]`

6.110.3 Member Function Documentation

6.110.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::createObject() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.110.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.110.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 303).

6.110.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 303).

6.110.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 304).

6.110.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 304).

```
6.110.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 305).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h`

6.111 `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTopicMarshaller` (p. 652).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.111.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p.652). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.111.2 Constructor & Destructor Documentation

6.111.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` [inline]

6.111.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` [inline, virtual]

6.111.3 Member Function Documentation

6.111.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.111.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.111.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 311).

6.111.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 311).

6.111.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 312).

6.111.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 312).

```
6.111.3.7 virtual void ac-
      tivemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightUnmarshal
      ( OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataInputStream * dataIn,
      utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 313).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h`

6.112 `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTopicMarshaller` (p. 656).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller`:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.112.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p.656). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.112.2 Constructor & Destructor Documentation

6.112.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller () [inline]`

6.112.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller () [inline, virtual]`

6.112.3 Member Function Documentation

6.112.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.112.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.112.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 307).

6.112.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 307).

6.112.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 308).

6.112.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 308).

```
6.112.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 309).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h`

6.113 `activemq::core::ActiveMQTransactionContext` Class Reference

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

```
#include <src/main/activemq/core/ActiveMQTransactionContext.h>
```

Public Member Functions

- **ActiveMQTransactionContext** (**ActiveMQSession** *session, const **decaf::util::Properties** &properties)
Constructor.
- virtual **~ActiveMQTransactionContext** ()
- virtual void **addSynchronization** (const **Pointer**< **Synchronization** > &sync)
*Adds a **Synchronization** (p. 3477) to this Transaction.*
- virtual void **removeSynchronization** (const **Pointer**< **Synchronization** > &sync)
*Removes a **Synchronization** (p. 3477) to this Transaction.*
- virtual void **begin** () throw (exceptions::ActiveMQException)
Begins a new transaction if one is not currently in progress.
- virtual void **commit** () throw (exceptions::ActiveMQException)
Commit the current Transaction.
- virtual void **rollback** () throw (exceptions::ActiveMQException)
Rollback the current Transaction.
- virtual const **decaf::lang::Pointer**< **commands::TransactionId** > &**getTransactionId** () const
Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.
- virtual bool **isInTransaction** () const
Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.

6.113.1 Detailed Description

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back. The Transaction represents an always running transaction, when it is committed or rolled back it silently creates a new transaction for the next set of messages. The only way to permanently end this transaction is to delete it.

Since

2.0

6.113.2 Constructor & Destructor Documentation

- 6.113.2.1** **activemq::core::ActiveMQTransactionContext::ActiveMQTransactionContext**
(**ActiveMQSession** * session, const **decaf::util::Properties** & properties)

Constructor.

Parameters

session The session that contains this transaction
properties Configuration parameters for this object

6.113.2.2 virtual
 activemq::core::ActiveMQTransactionContext::~~ActiveMQTransactionContext
 () [virtual]

6.113.3 Member Function Documentation

6.113.3.1 virtual void ac-
 tivemq::core::ActiveMQTransactionContext::addSynchronization (const
 Pointer< Synchronization > & *sync*) [virtual]

Adds a **Synchronization** (p. 3477) to this Transaction.

Parameters

sync - The **Synchronization** (p. 3477) instance to add.

6.113.3.2 virtual void activemq::core::ActiveMQTransactionContext::begin ()
 throw (exceptions::ActiveMQException) [virtual]

Begins a new transaction if one is not currently in progress.

Exceptions

ActiveMQException

6.113.3.3 virtual void activemq::core::ActiveMQTransactionContext::commit ()
 throw (exceptions::ActiveMQException) [virtual]

Commit the current Transaction.

Exceptions

ActiveMQException

6.113.3.4 virtual const decaf::lang::Pointer<commands::TransactionId>&
 activemq::core::ActiveMQTransactionContext::getTransactionId ()
 const [virtual]

Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.

Returns

TransactionInfo

Exceptions

InvalidStateException if a Transaction is not in progress.

6.113.3.5 `virtual bool activemq::core::ActiveMQTransactionContext::isInTransaction () const`
[virtual]

Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.

Returns

true if a transaction is in progress.

6.113.3.6 `virtual void activemq::core::ActiveMQTransactionContext::removeSynchronization (const Pointer< Synchronization > & sync)` [virtual]

Removes a **Synchronization** (p. 3477) to this Transaction.

Parameters

sync - The **Synchronization** (p. 3477) instance to add.

6.113.3.7 `virtual void activemq::core::ActiveMQTransactionContext::rollback ()`
`throw (exceptions::ActiveMQException)` [virtual]

Rollback the current Transaction.

Exceptions

ActiveMQException

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQTransactionContext.h`

6.114 decaf::util::zip::Adler32 Class Reference

Clas that can be used to compute an Adler-32 **Checksum** (p. 1059) for a data stream.

```
#include <src/main/decaf/util/zip/Adler32.h>
```

Inheritance diagram for decaf::util::zip::Adler32:

Public Member Functions

- **Adler32** ()
- virtual **~Adler32** ()
- virtual long long **getValue** () const
- virtual void **reset** ()

Reset the checksum to its initial value.

- virtual void **update** (const std::vector< unsigned char > &buffer)
Updates the current checksum with the specified vector of bytes.
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length)
throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Updates the current checksum with the specified array of bytes.
- virtual void **update** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Updates the current checksum with the specified array of bytes.
- virtual void **update** (int byte)
Updates the current checksum with the specified byte value.

6.114.1 Detailed Description

Clas that can be used to compute an Adler-32 **Checksum** (p.1059) for a data stream. The Alder-32 checksum trades reliability for speed over the CRC-32 algorithm.

Since

1.0

6.114.2 Constructor & Destructor Documentation

6.114.2.1 decaf::util::zip::Adler32::Adler32 ()

6.114.2.2 virtual decaf::util::zip::Adler32::~~Adler32 () [virtual]

6.114.3 Member Function Documentation

6.114.3.1 virtual long long decaf::util::zip::Adler32::getValue () const [virtual]

Returns

the current checksum value.

Implements **decaf::util::zip::Checksum** (p.1060).

6.114.3.2 virtual void decaf::util::zip::Adler32::reset () [virtual]

Reset the checksum to its initial value.

Implements **decaf::util::zip::Checksum** (p.1060).

6.114.3.3 virtual void decaf::util::zip::Adler32::update (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer The buffer to read the updated bytes from.

size The size of the passed buffer.

offset The position in the buffer to start reading.

length The amount of data to read from the byte buffer.

Exceptions

NullPointerException if the passed buffer is NULL.

IndexOutOfBoundsException if $\text{offset} + \text{length} > \text{size}$ of the buffer.

Implements **decaf::util::zip::Checksum** (p. 1061).

6.114.3.4 virtual void decaf::util::zip::Adler32::update (const std::vector< unsigned char > & *buffer*, int *offset*, int *length*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer The buffer to read the updated bytes from.

offset The position in the buffer to start reading.

length The amount of data to read from the byte buffer.

Exceptions

IndexOutOfBoundsException if $\text{offset} + \text{length} > \text{size}$ of the buffer.

Implements **decaf::util::zip::Checksum** (p. 1061).

6.114.3.5 virtual void decaf::util::zip::Adler32::update (const std::vector< unsigned char > & *buffer*) [virtual]

Updates the current checksum with the specified vector of bytes.

Parameters

buffer The buffer to read the updated bytes from.

Implements **decaf::util::zip::Checksum** (p. 1061).

6.114.3.6 virtual void decaf::util::zip::Adler32::update (int *byte*) [virtual]

Updates the current checksum with the specified byte value.

Parameters

- byte* The byte value to update the current **Checksum** (p. 1059) with (0..255).

Implements **decaf::util::zip::Checksum** (p. 1062).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Adler32.h**

6.115 decaf::lang::Appendable Class Reference

An object to which char sequences and values can be appended.

```
#include <src/main/decaf/lang/Appendable.h>
```

Inheritance diagram for decaf::lang::Appendable:

Public Member Functions

- virtual **~Appendable** ()
- virtual **Appendable** & **append** (char value)=0 throw (decaf::lang::Exception)
*Appends the specified character to this **Appendable** (p. 666).*
- virtual **Appendable** & **append** (const **CharSequence** *csq)=0 throw (decaf::lang::Exception)
*Appends the specified character sequence to this **Appendable** (p. 666).*
- virtual **Appendable** & **append** (const **CharSequence** *csq, int start, int end)=0 throw (decaf::lang::Exception)
*Appends a subsequence of the specified character sequence to this **Appendable** (p. 666).*

6.115.1 Detailed Description

An object to which char sequences and values can be appended. The **Appendable** (p. 666) interface must be implemented by any class whose instances are intended to receive formatted output from a Formatter.

TODO The characters to be appended should be valid Unicode characters as described in Unicode **Character** (p. 1019) Representation. Note that supplementary characters may be composed of multiple 16-bit char values.

Appendables are not necessarily safe for multithreaded access. **Thread** (p. 3520) safety is the responsibility of classes that extend and implement this interface.

Since this interface may be implemented by existing classes with different styles of error handling there is no guarantee that errors will be propagated to the invoker.

Since

1.0

6.115.2 Constructor & Destructor Documentation

6.115.2.1 virtual decaf::lang::Appendable::~~Appendable () [inline, virtual]

6.115.3 Member Function Documentation

6.115.3.1 virtual Appendable& decaf::lang::Appendable::append (char *value*)
throw (decaf::lang::Exception) [pure virtual]

Appends the specified character to this **Appendable** (p. 666).

Parameters

value The character to append.

Returns

a Reference to this **Appendable** (p. 666)

Exceptions

Exception (p. 1712) if an error occurs.

Implemented in **decaf::io::Writer** (p. 3757).

6.115.3.2 virtual Appendable& decaf::lang::Appendable::append (const
CharSequence * *csq*, int *start*, int *end*) throw (decaf::lang::Exception
) [pure virtual]

Appends a subsequence of the specified character sequence to this **Appendable** (p. 666).

Parameters

csq - The character sequence from which a subsequence will be appended. If *csq* is NULL, then characters will be appended as if *csq* contained the string "null".

start The index of the first character in the subsequence.

end The index of the character following the last character in the subsequence.

Returns

a Reference to this **Appendable** (p. 666)

Exceptions

Exception (p. 1712) if an error occurs.

IndexOutOfBoundsException *start* is greater than *end*, or *end* is greater than *csq.length()*

Implemented in **decaf::io::Writer** (p. 3758).

6.115.3.3 `virtual Appendable& decaf::lang::Appendable::append (const CharSequence * csq) throw (decaf::lang::Exception)` [pure virtual]

Appends the specified character sequence to this **Appendable** (p. 666).

Parameters

csq The character sequence from which a subsequence will be appended. If *csq* is NULL, then characters will be appended as if *csq* contained the string "null".

Returns

a Reference to this **Appendable** (p. 666).

Exceptions

Exception (p. 1712) if an error occurs.

Implemented in **decaf::io::Writer** (p. 3758).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Appendable.h`

6.116 decaf::internal::AprPool Class Reference

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

```
#include <src/main/decaf/internal/AprPool.h>
```

Public Member Functions

- **AprPool** ()
- virtual **~AprPool** ()
- `apr_pool_t * getAprPool ()` const
Gets the internal APR Pool.
- void **cleanup** ()
Clears data that was allocated by this pool.

Static Public Member Functions

- static `apr_pool_t * getGlobalPool ()`
Gets a pointer to the Global APR Pool used for the Application.

6.116.1 Detailed Description

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

6.116.2 Constructor & Destructor Documentation

6.116.2.1 `decaf::internal::AprPool::AprPool ()`

6.116.2.2 `virtual decaf::internal::AprPool::~~AprPool () [virtual]`

6.116.3 Member Function Documentation

6.116.3.1 `void decaf::internal::AprPool::cleanup ()`

Clears data that was allocated by this pool.

Users should call this after getting the data from the APR functions and copying it to someplace safe.

6.116.3.2 `apr_pool_t* decaf::internal::AprPool::getAprPool () const`

Gets the internal APR Pool.

Returns

the internal APR pool

6.116.3.3 `static apr_pool_t* decaf::internal::AprPool::getGlobalPool () [static]`

Gets a pointer to the Global APR Pool used for the Application.

Returns

pointer to the global APR Pool

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/AprPool.h`

6.117 decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 2756) that is a template on a Type and is **Thread** (p. 3520) Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

Data Structures

- `struct ArrayData`

Public Types

- typedef T * **PointerType**
- typedef T & **ReferenceType**
- typedef const T & **ConstReferenceType**
- typedef REFCOUNTER **CounterType**

Public Member Functions

- **ArrayPointer** ()
Default Constructor.
- **ArrayPointer** (int size)
*Create a new **ArrayPointer** (p. 669) instance and allocates an internal array that is sized using the passed in size value.*
- **ArrayPointer** (const **PointerType** value, int size)
*Explicit Constructor, creates an **ArrayPointer** (p. 669) that contains value with a single reference.*
- **ArrayPointer** (const **ArrayPointer** &value) throw ()
Copy constructor.
- virtual ~**ArrayPointer** () throw ()
- void **reset** (T *value, int size=0)
*Resets the **ArrayPointer** (p. 669) to hold the new value.*
- T * **release** ()
*Releases the **Pointer** (p. 2756) held and resets the internal pointer value to Null.*
- **PointerType** **get** () const
*Gets the real array pointer that is contained within this **Pointer** (p. 2756).*
- int **length** () const
Returns the current size of the contained array or zero if the array is NULL.
- void **swap** (**ArrayPointer** &value) throw ()
***Exception** (p. 1712) Safe Swap Function.*
- **ArrayPointer** **clone** () const
*Creates a new **ArrayPointer** (p. 669) instance that is a clone of the value contained in this **ArrayPointer** (p. 669).*
- **ArrayPointer** & **operator=** (const **ArrayPointer** &right) throw ()
*Assigns the value of right to this **Pointer** (p. 2756) and increments the reference Count.*
- template<typename T1 , typename R1 >
 ArrayPointer & **operator=** (const **ArrayPointer**< T1, R1 > &right) throw ()
- **ReferenceType** **operator[]** (int index)
Dereference Operator, returns a reference to the Contained value.

- **ConstReferenceType operator[]** (int index) const
- **bool operator!** () const
- **template<typename T1 , typename R1 >**
bool operator== (const **ArrayPointer**< T1, R1 > &right) const
- **template<typename T1 , typename R1 >**
bool operator!= (const **ArrayPointer**< T1, R1 > &right) const

Friends

- **bool operator==** (const **ArrayPointer** &left, const T *right)
- **bool operator==** (const T *left, const **ArrayPointer** &right)
- **bool operator!=** (const **ArrayPointer** &left, const T *right)
- **bool operator!=** (const T *left, const **ArrayPointer** &right)

6.117.1 Detailed Description

template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> class decaf::lang::ArrayPointer< T, REFCOUNTER >

Decaf's implementation of a Smart **Pointer** (p. 2756) that is a template on a Type and is **Thread** (p. 3520) Safe if the default Reference Counter is used. This **Pointer** (p. 2756) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of **ReferenceCounter**.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 2756) instances in the same manner as normal pointer, except that it does not provide an overload of operators (<, <=, >, >=). To allow use of a **Pointer** (p. 2756) in a STL container that requires it, **Pointer** (p. 2756) provides an implementation of std::less.

Since

1.0

6.117.2 Member Typedef Documentation

6.117.2.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> typedef const T& decaf::lang::ArrayPointer< T, REFCOUNTER >::ConstReferenceType`

6.117.2.2 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> typedef REFCOUNTER decaf::lang::ArrayPointer< T, REFCOUNTER >::CounterType`

6.117.2.3 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> typedef T* decaf::lang::ArrayPointer< T, REFCOUNTER >::PointerType`

6.117.2.4 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> typedef T& decaf::lang::ArrayPointer< T, REFCOUNTER >::ReferenceType`

6.117.3 Constructor & Destructor Documentation

6.117.3.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer () [inline]`

Default Constructor.

Initialized the contained array pointer to NULL, using the subscript operator results in an exception unless reset to contain a real value.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::clone()`, and `decaf::lang::ArrayPointer< unsigned char >::reset()`.

6.117.3.2 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer (int size) [inline]`

Create a new **ArrayPointer** (p. 669) instance and allocates an internal array that is sized using the passed in size value.

Parameters

size The size of the array to allocate for this **ArrayPointer** (p. 669) instance.

6.117.3.3 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer (const PointerType value, int size) [inline, explicit]`

Explicit Constructor, creates an **ArrayPointer** (p. 669) that contains value with a single reference. This object now has ownership until a call to release.

Parameters

value The pointer to the instance of the array we are taking ownership of.

size The size of the array this object is taking ownership of.

```
6.117.3.4  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount>
           decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer ( const
           ArrayPointer< T, REFCOUNTER > & value ) throw () [inline]
```

Copy constructor.

Copies the value contained in the **ArrayPointer** (p. 669) to the new instance and increments the reference counter.

```
6.117.3.5  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> virtual
           decaf::lang::ArrayPointer< T, REFCOUNTER >::~~ArrayPointer ( )
           throw () [inline, virtual]
```

6.117.4 Member Function Documentation

```
6.117.4.1  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> ArrayPointer
           decaf::lang::ArrayPointer< T, REFCOUNTER >::clone ( ) const
           [inline]
```

Creates a new **ArrayPointer** (p. 669) instance that is a clone of the value contained in this **ArrayPointer** (p. 669).

Returns

an **ArrayPointer** (p. 669) that contains a copy of the data in this **ArrayPointer** (p. 669).

```
6.117.4.2  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> PointerType
           decaf::lang::ArrayPointer< T, REFCOUNTER >::get ( ) const
           [inline]
```

Gets the real array pointer that is contained within this **Pointer** (p. 2756).

This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p. 2756). Use at your own risk.

Returns

the contained pointer.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::clone()`, `decaf::lang::ArrayPointerComparator< T, R >::compare()`, `decaf::lang::operator!=()`, `decaf::lang::ArrayPointer< unsigned char >::operator!=()`, `std::less< decaf::lang::ArrayPointer< T > >::operator()()`, `decaf::lang::ArrayPointerComparator< T, R >::operator()()`, `decaf::lang::operator==()`, and `decaf::lang::ArrayPointer< unsigned char >::operator==()`.

6.117.4.3 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> int
decaf::lang::ArrayPointer< T, REFCOUNTER >::length () const
[inline]`

Returns the current size of the contained array or zero if the array is NULL.

Returns

the size of the array or zero if the array is NULL

6.117.4.4 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> bool
decaf::lang::ArrayPointer< T, REFCOUNTER >::operator! () const
[inline]`

6.117.4.5 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename
T1 , typename R1 > bool decaf::lang::ArrayPointer< T, REFCOUNTER
>::operator!= (const ArrayPointer< T1, R1 > & right) const
[inline]`

6.117.4.6 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename
T1 , typename R1 > ArrayPointer& decaf::lang::ArrayPointer< T,
REFCOUNTER >::operator= (const ArrayPointer< T1, R1 > & right
) throw () [inline]`

6.117.4.7 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> ArrayPointer&
decaf::lang::ArrayPointer< T, REFCOUNTER >::operator= (const
ArrayPointer< T, REFCOUNTER > & right) throw () [inline]`

Assigns the value of *right* to this **Pointer** (p. 2756) and increments the reference Count.

Parameters

right - **Pointer** (p. 2756) on the right hand side of an operator= call to this.

6.117.4.8 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename
T1 , typename R1 > bool decaf::lang::ArrayPointer< T, REFCOUNTER
>::operator== (const ArrayPointer< T1, R1 > & right) const
[inline]`

6.117.4.9 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> ReferenceType
decaf::lang::ArrayPointer< T, REFCOUNTER >::operator[] (int index
) [inline]`

Dereference Operator, returns a reference to the Contained value.

This method throws an `NullPointerException` if the contained value is NULL.

Returns

reference to the contained pointer.

Exceptions

NullPointerException if the contained value is Null

```
6.117.4.10  template<typename T, typename REFCOUNTER
            = decaf::util::concurrent::atomic::AtomicRefCounter>
            ConstReferenceType decaf::lang::ArrayPointer< T, REFCOUNTER
            >::operator[] ( int index ) const [inline]
```

```
6.117.4.11  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> T*
            decaf::lang::ArrayPointer< T, REFCOUNTER >::release (    ) [inline]
```

Releases the **Pointer** (p. 2756) held and resets the internal pointer value to Null.

This method is not guaranteed to be safe if the **Pointer** (p. 2756) is held by more than one object or this method is called from more than one thread.

Parameters

value - The new value to contain.

Returns

The pointer instance that was held by this **Pointer** (p. 2756) object, the pointer is no longer owned by this **Pointer** (p. 2756) and won't be freed when this **Pointer** (p. 2756) goes out of scope.

Referenced by decaf::lang::ArrayPointer< unsigned char >::ArrayPointer().

```
6.117.4.12  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> void
            decaf::lang::ArrayPointer< T, REFCOUNTER >::reset ( T * value,
            int size = 0 ) [inline]
```

Resets the **ArrayPointer** (p. 669) to hold the new value.

Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p. 2756) to a NULL pointer.

Parameters

value The new array pointer value to contain.

size The size of the new array value this object now contains.

```
6.117.4.13  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> void
            decaf::lang::ArrayPointer< T, REFCOUNTER >::swap (
            ArrayPointer< T, REFCOUNTER > & value ) throw () [inline]
```

Exception (p. 1712) Safe Swap Function.

Parameters

value - the value to swap with this.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::operator=()`, and `decaf::lang::ArrayPointer< unsigned char >::swap()`.

6.117.5 Friends And Related Function Documentation

6.117.5.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> bool operator!= (const ArrayPointer< T, REFCOUNTER > & left, const T * right)`
[friend]

6.117.5.2 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> bool operator!= (const T * left, const ArrayPointer< T, REFCOUNTER > & right)`
[friend]

6.117.5.3 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> bool operator== (const ArrayPointer< T, REFCOUNTER > & left, const T * right)`
[friend]

6.117.5.4 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> bool operator== (const T * left, const ArrayPointer< T, REFCOUNTER > & right)`
[friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/ArrayPointer.h`

6.118 decaf::lang::ArrayPointerComparator< T, R > Class Template Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p.669).

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

Inheritance diagram for `decaf::lang::ArrayPointerComparator< T, R >`:

Public Member Functions

- virtual bool **operator()** (const **ArrayPointer**< T, R > &*left*, const **ArrayPointer**< T, R > &*right*) const
- virtual int **compare** (const **ArrayPointer**< T, R > &*left*, const **ArrayPointer**< T, R > &*right*) const

6.118.1 Detailed Description

```
template<typename T, typename R = decaf::util::concurrent::atomic::AtomicRefCounter>
class decaf::lang::ArrayPointerComparator< T, R >
```

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 669). This allows for a basic ordering to be acheived in Decaf containers.

Custom implementations are possible where an array of some type has a logical natural ordering such as array of integers where the sum of all ints in the array is used.

6.118.2 Member Function Documentation

6.118.2.1

```
template<typename T , typename R =
decaf::util::concurrent::atomic::AtomicRefCounter> virtual
int decaf::lang::ArrayPointerComparator< T, R >::compare ( const
ArrayPointer< T, R > & left, const ArrayPointer< T, R > & right )
const [inline, virtual]
```

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

6.118.2.2

```
template<typename T , typename R =
decaf::util::concurrent::atomic::AtomicRefCounter> virtual
bool decaf::lang::ArrayPointerComparator< T, R >::operator() ( const
ArrayPointer< T, R > & left, const ArrayPointer< T, R > & right )
const [inline, virtual]
```

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**ArrayPointer.h**

6.119 decaf::util::concurrent::atomic::AtomicBoolean Class Reference

A boolean value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicBoolean.h>
```

Public Member Functions

- **AtomicBoolean** ()
*Creates a new **AtomicBoolean** (p. 677) whose initial value is false.*
- **AtomicBoolean** (bool initialValue)
*Creates a new **AtomicBoolean** (p. 677) with the initial value.*
- virtual ~**AtomicBoolean** ()

- `bool get () const`
*Gets the current value of this **AtomicBoolean** (p. 677).*
- `void set (bool newValue)`
Unconditionally sets to the given value.
- `bool compareAndSet (bool expect, bool update)`
Atomically sets the value to the given updated value if the current value == the expected value.
- `bool getAndSet (bool newValue)`
Atomically sets to the given value and returns the previous value.
- `std::string toString () const`
Returns the String representation of the current value.

6.119.1 Detailed Description

A boolean value that may be updated atomically. An **AtomicBoolean** (p. 677) is used in applications such as atomically updated flags, and cannot be used as a replacement for a `Boolean`.

6.119.2 Constructor & Destructor Documentation

6.119.2.1 `decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean ()`

Creates a new **AtomicBoolean** (p. 677) whose initial value is false.

6.119.2.2 `decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean (bool initialValue)`

Creates a new **AtomicBoolean** (p. 677) with the initial value.

Parameters

initialValue - The initial value of this boolean.

6.119.2.3 `virtual decaf::util::concurrent::atomic::AtomicBoolean::~~AtomicBoolean () [inline, virtual]`

6.119.3 Member Function Documentation

6.119.3.1 `bool decaf::util::concurrent::atomic::AtomicBoolean::compareAndSet (bool expect, bool update)`

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters

expect - the expected value

update - the new value

Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

6.119.3.2 `bool decaf::util::concurrent::atomic::AtomicBoolean::get () const` [inline]

Gets the current value of this **AtomicBoolean** (p.677).

Returns

the currently set value.

6.119.3.3 `bool decaf::util::concurrent::atomic::AtomicBoolean::getAndSet (bool new Value)`

Atomically sets to the given value and returns the previous value.

Parameters

new Value - the new value

Returns

the previous value

6.119.3.4 `void decaf::util::concurrent::atomic::AtomicBoolean::set (bool new Value)` [inline]

Unconditionally sets to the given value.

Parameters

new Value - the new value

6.119.3.5 `std::string decaf::util::concurrent::atomic::AtomicBoolean::toString () const`

Returns the String representation of the current value.

Returns

the String representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicBoolean.h`

6.120 decaf::util::concurrent::atomic::AtomicInteger Class Reference

An int value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicInteger.h>
```

Inheritance diagram for decaf::util::concurrent::atomic::AtomicInteger:

Public Member Functions

- **AtomicInteger** ()
*Create a new **AtomicInteger** (p. 680) with an initial value of 0.*
- **AtomicInteger** (int initialValue)
*Create a new **AtomicInteger** (p. 680) with the given initial value.*
- virtual **~AtomicInteger** ()
- int **get** () const
Gets the current value.
- void **set** (int newValue)
Sets to the given value.
- int **getAndSet** (int newValue)
Atomically sets to the given value and returns the old value.
- bool **compareAndSet** (int expect, int update)
Atomically sets the value to the given updated value if the current value == the expected value.
- int **getAndIncrement** ()
Atomically increments by one the current value.
- int **getAndDecrement** ()
Atomically decrements by one the current value.
- int **getAndAdd** (int delta)
Atomically adds the given value to the current value.
- int **incrementAndGet** ()
Atomically increments by one the current value.
- int **decrementAndGet** ()
Atomically decrements by one the current value.
- int **addAndGet** (int delta)
Atomically adds the given value to the current value.

- `std::string toString () const`
Returns the String representation of the current value.
- `int intValue () const`
Description copied from class: Number Returns the value of the specified number as an int.
- `long long longValue () const`
Description copied from class: Number Returns the value of the specified number as a long.
- `float floatValue () const`
Description copied from class: Number Returns the value of the specified number as a float.
- `double doubleValue () const`
Description copied from class: Number Returns the value of the specified number as a double.

6.120.1 Detailed Description

An int value that may be updated atomically. An **AtomicInteger** (p. 680) is used in applications such as atomically incremented counters, and cannot be used as a replacement for an Integer. However, this class does extend Number to allow uniform access by tools and utilities that deal with numerically-based classes.

6.120.2 Constructor & Destructor Documentation

6.120.2.1 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger ()

Create a new **AtomicInteger** (p. 680) with an initial value of 0.

6.120.2.2 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger (int *initialValue*)

Create a new **AtomicInteger** (p. 680) with the given initial value.

Parameters

initialValue - The initial value of this object.

6.120.2.3 virtual decaf::util::concurrent::atomic::AtomicInteger::~~AtomicInteger () [inline, virtual]

6.120.3 Member Function Documentation

6.120.3.1 int decaf::util::concurrent::atomic::AtomicInteger::addAndGet (int *delta*)

Atomically adds the given value to the current value.

Parameters

delta - the value to add.

Returns

the updated value.

6.120.3.2 `bool decaf::util::concurrent::atomic::AtomicInteger::compareAndSet (int expect, int update)`

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters

expect - the expected value

update - the new value

Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

6.120.3.3 `int decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet ()`

Atomically decrements by one the current value.

Returns

the updated value.

Referenced by `decaf::util::concurrent::atomic::AtomicRefCounter::release()`.

6.120.3.4 `double decaf::util::concurrent::atomic::AtomicInteger::doubleValue () const [virtual]`

Description copied from class: Number Returns the value of the specified number as a double.

This may involve rounding.

Returns

the numeric value represented by this object after conversion to type double.

Implements `decaf::lang::Number` (p. 2654).

6.120.3.5 `float decaf::util::concurrent::atomic::AtomicInteger::floatValue () const [virtual]`

Description copied from class: Number Returns the value of the specified number as a float.

This may involve rounding.

Returns

the numeric value represented by this object after conversion to type float.

Implements **decaf::lang::Number** (p. 2654).

6.120.3.6 `int decaf::util::concurrent::atomic::AtomicInteger::get () const`
[inline]

Gets the current value.

Returns

the current value.

6.120.3.7 `int decaf::util::concurrent::atomic::AtomicInteger::getAndAdd (int`
`delta)`

Atomically adds the given value to the current value.

Parameters

delta - The value to add.

Returns

the previous value.

6.120.3.8 `int decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement (`
`)`

Atomically decrements by one the current value.

Returns

the previous value.

6.120.3.9 `int decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement ()`

Atomically increments by one the current value.

Returns

the previous value.

6.120.3.10 `int decaf::util::concurrent::atomic::AtomicInteger::getAndSet (int`
`new Value)`

Atomically sets to the given value and returns the old value.

Parameters

new Value - the new value.

Returns

the previous value.

6.120.3.11 `int decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet ()`

Atomically increments by one the current value.

Returns

the updated value.

Referenced by `decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter()`.

6.120.3.12 `int decaf::util::concurrent::atomic::AtomicInteger::intValue () const`
[virtual]

Description copied from class: Number Returns the value of the specified number as an int.

This may involve rounding or truncation.

Returns

the numeric value represented by this object after conversion to type int.

Implements **decaf::lang::Number** (p. 2654).

6.120.3.13 `long long decaf::util::concurrent::atomic::AtomicInteger::longValue ()`
`const` [virtual]

Description copied from class: Number Returns the value of the specified number as a long.

This may involve rounding or truncation.

Returns

the numeric value represented by this object after conversion to type long long.

Implements **decaf::lang::Number** (p. 2654).

6.120.3.14 `void decaf::util::concurrent::atomic::AtomicInteger::set (int new Value)`
[inline]

Sets to the given value.

Parameters

new Value - the new value

6.120.3.15 std::string decaf::util::concurrent::atomic::AtomicInteger::toString () const

Returns the String representation of the current value.

Returns

the String representation of the current value.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/**AtomicInteger.h**

6.121 decaf::util::concurrent::atomic::AtomicRefCounter Class Reference

```
#include <src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h>
```

Inherited by decaf::lang::ArrayPointer< unsigned char >, decaf::lang::Pointer< activemq::threads::TaskRunner >, decaf::lang::Pointer< ActiveMQDestination >, decaf::lang::Pointer< ActiveMQTransactionContext >, decaf::lang::Pointer< BackupTransportPool >, decaf::lang::Pointer< BooleanExpression >, decaf::lang::Pointer< BrokerError >, decaf::lang::Pointer< BrokerId >, decaf::lang::Pointer< ByteArrayAdapter >, decaf::lang::Pointer< CloseTransportsTask >, decaf::lang::Pointer< cms::Destination >, decaf::lang::Pointer< commands::ActiveMQDestination >, decaf::lang::Pointer< commands::ConsumerId >, decaf::lang::Pointer< commands::ConsumerInfo >, decaf::lang::Pointer< commands::Message >, decaf::lang::Pointer< commands::MessageAck >, decaf::lang::Pointer< commands::ProducerInfo >, decaf::lang::Pointer< commands::SessionInfo >, decaf::lang::Pointer< commands::TransactionId >, decaf::lang::Pointer< commands::WireFormatInfo >, decaf::lang::Pointer< Comparator< E > >, decaf::lang::Pointer< CompositeTaskRunner >, decaf::lang::Pointer< ConnectionId >, decaf::lang::Pointer< ConnectionInfo >, decaf::lang::Pointer< ConsumerId >, decaf::lang::Pointer< ConsumerInfo >, decaf::lang::Pointer< core::ActiveMQAckHandler >, decaf::lang::Pointer< DataStructure >, decaf::lang::Pointer< decaf::lang::Runnable >, decaf::lang::Pointer< decaf::lang::Thread >, decaf::lang::Pointer< Exception >, decaf::lang::Pointer< LockHandle >, decaf::lang::Pointer< LogManagerInternals >, decaf::lang::Pointer< Message >, decaf::lang::Pointer< MessageAck >, decaf::lang::Pointer< MessageId >, decaf::lang::Pointer< ProducerId >, decaf::lang::Pointer< ProducerInfo >, decaf::lang::Pointer< Properties >, decaf::lang::Pointer< Response >, decaf::lang::Pointer< ResponseBuilder >, decaf::lang::Pointer< SessionId >, decaf::lang::Pointer< SessionInfo >, decaf::lang::Pointer< Tracked >, decaf::lang::Pointer< TransactionId >, decaf::lang::Pointer< TransactionState >, decaf::lang::Pointer< Transport >, decaf::lang::Pointer< TransportListener >, decaf::lang::Pointer< URI >, decaf::lang::Pointer< URIPool >, and decaf::lang::Pointer< wireformat::WireFormat >.

Public Member Functions

- AtomicRefCounter ()
- AtomicRefCounter (const AtomicRefCounter &other)

- virtual `~AtomicRefCounter ()`

Protected Member Functions

- void `swap (AtomicRefCounter &other)`

Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

- bool `release ()`

Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.

6.121.1 Constructor & Destructor Documentation

6.121.1.1 `decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter ()` [inline]

6.121.1.2 `decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter (const AtomicRefCounter & other)` [inline]

References `decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet()`.

6.121.1.3 virtual `decaf::util::concurrent::atomic::AtomicRefCounter::~~AtomicRefCounter ()` [inline, virtual]

6.121.2 Member Function Documentation

6.121.2.1 bool `decaf::util::concurrent::atomic::AtomicRefCounter::release ()` [inline, protected]

Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.

Returns

true if the count is now zero.

Reimplemented in `decaf::lang::ArrayPointer< unsigned char >` (p. 675), `decaf::lang::Pointer< MessageAck >` (p. 2762), `decaf::lang::Pointer< BooleanExpression >` (p. 2762), `decaf::lang::Pointer< commands::ConsumerId >` (p. 2762), `decaf::lang::Pointer< BrokerError >` (p. 2762), `decaf::lang::Pointer< Transport >` (p. 2762), `decaf::lang::Pointer< wireformat::WireFormat >` (p. 2762), `decaf::lang::Pointer< commands::WireFormatInfo >` (p. 2762), `decaf::lang::Pointer< CloseTransportsTask >` (p. 2762), `decaf::lang::Pointer< CompositeTaskRunner >` (p. 2762), `decaf::lang::Pointer< ActiveMQTransactionContext >` (p. 2762), `decaf::lang::Pointer< commands::ProducerInfo >` (p. 2762), `decaf::lang::Pointer< Comparator< E > >` (p. 2762), `decaf::lang::Pointer< BrokerId >` (p. 2762), `decaf::lang::Pointer< LogManagerInternals >` (p. 2762), `decaf::lang::Pointer<`

commands::SessionInfo > (p. 2762), decaf::lang::Pointer< Message > (p. 2762),
 decaf::lang::Pointer< URI > (p. 2762), decaf::lang::Pointer< DataStructure
 > (p. 2762), decaf::lang::Pointer< activemq::threads::TaskRunner > (p. 2762),
 decaf::lang::Pointer< LockHandle > (p. 2762), decaf::lang::Pointer< com-
 mands::ActiveMQDestination > (p. 2762), decaf::lang::Pointer< ConsumerInfo
 > (p. 2762), decaf::lang::Pointer< ConnectionId > (p. 2762), decaf::lang::Pointer<
 decaf::lang::Runnable > (p. 2762), decaf::lang::Pointer< Properties > (p. 2762),
 decaf::lang::Pointer< BackupTransportPool > (p. 2762), decaf::lang::Pointer<
 ProducerInfo > (p. 2762), decaf::lang::Pointer< decaf::lang::Thread > (p. 2762),
 decaf::lang::Pointer< MessageId > (p. 2762), decaf::lang::Pointer< Response
 > (p. 2762), decaf::lang::Pointer< SessionId > (p. 2762), decaf::lang::Pointer<
 cms::Destination > (p. 2762), decaf::lang::Pointer< TransportListener > (p. 2762),
 decaf::lang::Pointer< commands::TransactionId > (p. 2762), decaf::lang::Pointer<
 ActiveMQDestination > (p. 2762), decaf::lang::Pointer< ProducerId > (p. 2762),
 decaf::lang::Pointer< ResponseBuilder > (p. 2762), decaf::lang::Pointer< SessionInfo >
 (p. 2762), decaf::lang::Pointer< commands::Message > (p. 2762), decaf::lang::Pointer<
 Tracked > (p. 2762), decaf::lang::Pointer< ConnectionInfo > (p. 2762),
 decaf::lang::Pointer< commands::MessageAck > (p. 2762), decaf::lang::Pointer<
 core::ActiveMQAckHandler > (p. 2762), decaf::lang::Pointer< Exception > (p. 2762),
 decaf::lang::Pointer< TransactionState > (p. 2762), decaf::lang::Pointer< com-
 mands::ConsumerInfo > (p. 2762), decaf::lang::Pointer< ConsumerId > (p. 2762),
 decaf::lang::Pointer< URIPool > (p. 2762), decaf::lang::Pointer< ByteArrayAdapter
 > (p. 2762), and decaf::lang::Pointer< TransactionId > (p. 2762).

References decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet().

6.121.2.2 void decaf::util::concurrent::atomic::AtomicRefCounter::swap (AtomicRefCounter & *other*) [inline, protected]

Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

Parameters

other The value to swap with this one's.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h

6.122 decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference

An Pointer reference that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicReference.h>
```

Public Member Functions

- AtomicReference ()
- AtomicReference (T *value)

- virtual `~AtomicReference ()`
- `T * get () const`
Gets the Current Value.
- void `set (T *newValue)`
Sets the Current value of this Reference.
- bool `compareAndSet (T *expect, T *update)`
Atomically sets the value to the given updated value if the current value == the expected value.
- `T * getAndSet (T *newValue)`
Atomically sets to the given value and returns the old value.
- `std::string toString () const`
Returns the String representation of the current value.

6.122.1 Detailed Description

`template<typename T> class decaf::util::concurrent::atomic::AtomicReference< T >`

An Pointer reference that may be updated atomically.

6.122.2 Constructor & Destructor Documentation

6.122.2.1 `template<typename T >`
`decaf::util::concurrent::atomic::AtomicReference< T`
`>::AtomicReference () [inline]`

6.122.2.2 `template<typename T >`
`decaf::util::concurrent::atomic::AtomicReference< T`
`>::AtomicReference (T * value) [inline]`

6.122.2.3 `template<typename T > virtual`
`decaf::util::concurrent::atomic::AtomicReference< T`
`>::~AtomicReference () [inline, virtual]`

6.122.3 Member Function Documentation

6.122.3.1 `template<typename T > bool`
`decaf::util::concurrent::atomic::AtomicReference< T`
`>::compareAndSet (T * expect, T * update) [inline]`

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters

expect - the expected value

update - the new value

Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

6.122.3.2 `template<typename T > T*
decaf::util::concurrent::atomic::AtomicReference< T >::get
() const [inline]`

Gets the Current Value.

Returns

the current value of this Reference.

6.122.3.3 `template<typename T > T*
decaf::util::concurrent::atomic::AtomicReference< T
>::getAndSet (T * new Value) [inline]`

Atomically sets to the given value and returns the old value.

Parameters

new Value- the new value

Returns

the previous value.

6.122.3.4 `template<typename T > void
decaf::util::concurrent::atomic::AtomicReference< T >::set
(T * new Value) [inline]`

Sets the Current value of this Reference.

Parameters

new Value The new Value of this Reference.

6.122.3.5 `template<typename T > std::string
decaf::util::concurrent::atomic::AtomicReference< T
>::toString () const [inline]`

Returns the String representation of the current value.

Returns

string representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicReference.h`

6.123 activemq::transport::failover::BackupTransport Class Reference

#include <src/main/activemq/transport/failover/BackupTransport.h>

Inheritance diagram for activemq::transport::failover::BackupTransport:

Public Member Functions

- **BackupTransport** (**BackupTransportPool** *failover)
- virtual **~BackupTransport** ()
- **decaf::net::URI** **getUri** () const
Gets the URI assigned to this Backup.
- void **setUri** (const **decaf::net::URI** &uri)
*Sets the URI assigned to this **Transport** (p. 3629).*
- const **Pointer**< **Transport** > & **getTransport** ()
Gets the currently held transport.
- void **setTransport** (const **Pointer**< **Transport** > &transport)
Sets the held transport, if not NULL then start to listen for exceptions from the held transport.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- bool **isClosed** () const
*Has the **Transport** (p. 3629) been shutdown and no longer usable.*
- void **setClosed** (bool value)
*Sets the closed flag on this **Transport** (p. 3629).*

6.123.1 Constructor & Destructor Documentation

6.123.1.1 **activemq::transport::failover::BackupTransport::BackupTransport** (**BackupTransportPool** * *failover*)

6.123.1.2 virtual **activemq::transport::failover::BackupTransport::~~BackupTransport** ()
[virtual]

6.123.2 Member Function Documentation

6.123.2.1 const **Pointer**<**Transport**>& **activemq::transport::failover::BackupTransport::getTransport** () [inline]

Gets the currently held transport.

Returns

pointer to the held transport or NULL if not set.

6.123.2.2 decaf::net::URI activemq::transport::failover::BackupTransport::getUri () const [inline]

Gets the URI assigned to this Backup.

Returns

the assigned URI

6.123.2.3 bool activemq::transport::failover::BackupTransport::isClosed () const [inline]

Has the **Transport** (p. 3629) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3629)

6.123.2.4 virtual void activemq::transport::failover::BackupTransport::onException (const decaf::lang::Exception & *ex*) [virtual]

Event handler for an exception from a command transport.

The **BackupTransport** (p. 690) closes its internal **Transport** (p. 3629) when an exception is received and returns the URI to the pool of URIs to attempt connections to.

Parameters

ex The exception that was passed to this listener to handle.

Implements **activemq::transport::TransportListener** (p. 3645).

6.123.2.5 void activemq::transport::failover::BackupTransport::setClosed (bool *value*) [inline]

Sets the closed flag on this **Transport** (p. 3629).

Parameters

value - true for closed.

6.123.2.6 void activemq::transport::failover::BackupTransport::setTransport (const Pointer< Transport > & *transport*) [inline]

Sets the held transport, if not NULL then start to listen for exceptions from the held transport.

Parameters

transport The transport to hold.

6.123.2.7 void activemq::transport::failover::BackupTransport::setUri (const decaf::net::URI & uri) [inline]

Sets the URI assigned to this **Transport** (p. 3629).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/BackupTransport.h

6.124 activemq::transport::failover::BackupTransportPool Class Reference

```
#include <src/main/activemq/transport/failover/BackupTransportPool.h>
```

Inheritance diagram for activemq::transport::failover::BackupTransportPool:

Public Member Functions

- **BackupTransportPool** (const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- **BackupTransportPool** (int backupPoolSize, const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- virtual ~**BackupTransportPool** ()
- virtual bool **isPending** () const
Return true if we don't currently have enough Connected Transports.
- **Pointer**< **BackupTransport** > **getBackup** ()
*Get a Connected **Transport** (p. 3629) from the pool of Backups if any are present, otherwise it return a NULL Pointer.*
- virtual bool **iterate** ()
Connect to a Backup Broker if we haven't already connected to the max number of Backups.
- int **getBackupPoolSize** () const
Gets the Max number of Backups this Task will create.
- void **setBackupPoolSize** (int size)
Sets the Max number of Backups this Task will create.
- bool **isEnabled** () const
*Gets if the backup **Transport** (p. 3629) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.*
- void **setEnabled** (bool value)
*Sets if this Backup **Transport** (p. 3629) Pool is enabled.*

Friends

- class **BackupTransport**

6.124.1 Constructor & Destructor Documentation

6.124.1.1 `activemq::transport::failover::BackupTransportPool::BackupTransportPool (const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool)`

6.124.1.2 `activemq::transport::failover::BackupTransportPool::BackupTransportPool (int backupPoolSize, const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool)`

6.124.1.3 `virtual
activemq::transport::failover::BackupTransportPool::~~BackupTransportPool () [virtual]`

6.124.2 Member Function Documentation

6.124.2.1 `Pointer<BackupTransport> activemq::transport::failover::BackupTransportPool::getBackup ()`

Get a Connected **Transport** (p. 3629) from the pool of Backups if any are present, otherwise it return a NULL Pointer.

Returns

Pointer to a Connected **Transport** (p. 3629) or NULL

6.124.2.2 `int activemq::transport::failover::BackupTransportPool::getBackupPoolSize () const [inline]`

Gets the Max number of Backups this Task will create.

Returns

the max number of active BackupTransports that will be created.

6.124.2.3 `bool activemq::transport::failover::BackupTransportPool::isEnabled () const [inline]`

Gets if the backup **Transport** (p. 3629) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.

Returns

true if enable.

6.124.2.4 `virtual bool activemq::transport::failover::BackupTransportPool::isPending () const` [virtual]

Return true if we don't currently have enough Connected Transports.

Implements `activemq::threads::CompositeTask` (p. 1132).

6.124.2.5 `virtual bool activemq::transport::failover::BackupTransportPool::iterate ()` [virtual]

Connect to a Backup Broker if we haven't already connected to the max number of Backups.

Implements `activemq::threads::Task` (p. 3495).

6.124.2.6 `void activemq::transport::failover::BackupTransportPool::setBackupPoolSize (int size)` [inline]

Sets the Max number of Backups this Task will create.

Parameters

size - the max number of active BackupTransports that will be created.

6.124.2.7 `void activemq::transport::failover::BackupTransportPool::setEnabled (bool value)`

Sets if this Backup **Transport** (p. 3629) Pool is enabled.

When not enabled no Backups are created and any that were are destroyed.

Parameters

value - true to enable backup creation, false to disable.

6.124.3 Friends And Related Function Documentation

6.124.3.1 `friend class BackupTransport` [friend]

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/BackupTransportPool.h`

6.125 activemq::commands::BaseCommand Class Reference

```
#include <src/main/activemq/commands/BaseCommand.h>
```

Inheritance diagram for `activemq::commands::BaseCommand`:

Public Member Functions

- **BaseCommand** ()
- virtual **~BaseCommand** ()
- virtual void **setCommandId** (int id)
*Sets the **Command** (p. 1107) Id of this **Message** (p. 2358).*
- virtual int **getCommandId** () const
*Gets the **Command** (p. 1107) Id of this **Message** (p. 2358).*
- virtual void **setResponseRequired** (const bool required)
*Set if this **Message** (p. 2358) requires a **Response** (p. 3076).*
- virtual bool **isResponseRequired** () const
*Is a **Response** (p. 3076) required for this **Command** (p. 1107).*
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual bool **isConnectionInfo** () const
- virtual bool **isConsumerInfo** () const
- virtual bool **isBrokerInfo** () const
- virtual bool **isMessage** () const
- virtual bool **isMessageAck** () const
- virtual bool **isKeepAliveInfo** () const
- virtual bool **isMessageDispatch** () const
- virtual bool **isMessageDispatchNotification** () const
- virtual bool **isProducerAck** () const
- virtual bool **isProducerInfo** () const
- virtual bool **isResponse** () const
- virtual bool **isRemoveInfo** () const
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual bool **isShutdownInfo** () const
- virtual bool **isTransactionInfo** () const
- virtual bool **isWireFormatInfo** () const

6.125.1 Constructor & Destructor Documentation

6.125.1.1 `activemq::commands::BaseCommand::BaseCommand ()` [inline]

6.125.1.2 `virtual activemq::commands::BaseCommand::~~BaseCommand ()`
[inline, virtual]

6.125.2 Member Function Documentation

6.125.2.1 `virtual void activemq::commands::BaseCommand::copyDataStructure (const DataStructure * src)` [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

`src` - Source Object

Implements `activemq::commands::DataStructure` (p. 1555).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 168), `activemq::commands::ActiveMQBytesMessage` (p. 198), `activemq::commands::ActiveMQMapMessage` (p. 320), `activemq::commands::ActiveMQMessage` (p. 354), `activemq::commands::ActiveMQObjectMessage` (p. 398), `activemq::commands::ActiveMQStreamMessage` (p. 489), `activemq::commands::ActiveMQTextMessage` (p. 607), `activemq::commands::BrokerError` (p. 794), `activemq::commands::BrokerInfo` (p. 826), `activemq::commands::ConnectionControl` (p. 1173), `activemq::commands::ConnectionError` (p. 1202), `activemq::commands::ConnectionInfo` (p. 1259), `activemq::commands::ConsumerControl` (p. 1304), `activemq::commands::ConsumerInfo` (p. 1360), `activemq::commands::ControlCommand` (p. 1391), `activemq::commands::DataArrayResponse` (p. 1424), `activemq::commands::DataResponse` (p. 1478), `activemq::commands::DestinationInfo` (p. 1615), `activemq::commands::ExceptionResponse` (p. 1721), `activemq::commands::FlushCommand` (p. 1813), `activemq::commands::IntegerResponse` (p. 1957), `activemq::commands::KeepAliveInfo` (p. 2124), `activemq::commands::Message` (p. 2363), `activemq::commands::MessageAck` (p. 2395), `activemq::commands::MessageDispatch` (p. 2428), `activemq::commands::MessageDispatchNotification` (p. 2463), `activemq::commands::MessagePull` (p. 2565), `activemq::commands::ProducerAck` (p. 2841), `activemq::commands::ProducerInfo` (p. 2899), `activemq::commands::RemoveInfo` (p. 2989), `activemq::commands::RemoveSubscriptionInfo` (p. 3016), `activemq::commands::ReplayCommand` (p. 3044), `activemq::commands::Response` (p. 3078), `activemq::commands::SessionInfo` (p. 3190), `activemq::commands::ShutdownInfo` (p. 3251), `activemq::commands::TransactionInfo` (p. 3596), and `activemq::commands::WireFormatInfo` (p. 3721).

References `getCommandId()`, and `isResponseRequired()`.

6.125.2.2 `virtual bool activemq::commands::BaseCommand::equals (const DataStructure * value) const` [inline, virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1556).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 168), `activemq::commands::ActiveMQBytesMessage` (p. 199), `activemq::commands::ActiveMQMapMessage` (p. 320), `activemq::commands::ActiveMQMessage` (p. 354), `activemq::commands::ActiveMQMessageTemplate< T >` (p. 383), `activemq::commands::ActiveMQObjectMessage` (p. 398), `activemq::commands::ActiveMQStreamMessage` (p. 490), `activemq::commands::ActiveMQTextMessage` (p. 607), `activemq::commands::BrokerInfo` (p. 827), `activemq::commands::ConnectionControl` (p. 1174), `activemq::commands::ConnectionError` (p. 1202), `activemq::commands::ConnectionInfo` (p. 1260), `activemq::commands::ConsumerControl` (p. 1304), `activemq::commands::ConsumerInfo` (p. 1360), `activemq::commands::ControlCommand` (p. 1392), `activemq::commands::DataArrayResponse` (p. 1424), `activemq::commands::DataResponse` (p. 1478), `activemq::commands::DestinationInfo` (p. 1615), `activemq::commands::ExceptionResponse` (p. 1721), `activemq::commands::FlushCommand` (p. 1813), `activemq::commands::IntegerResponse` (p. 1957), `activemq::commands::KeepAliveInfo` (p. 2124), `activemq::commands::Message` (p. 2363), `activemq::commands::MessageAck` (p. 2396), `activemq::commands::MessageDispatch` (p. 2428), `activemq::commands::MessageDispatchNotification` (p. 2464), `activemq::commands::MessagePull` (p. 2565), `activemq::commands::ProducerAck` (p. 2841), `activemq::commands::ProducerInfo` (p. 2899), `activemq::commands::RemoveInfo` (p. 2989), `activemq::commands::RemoveSubscriptionInfo` (p. 3017), `activemq::commands::ReplayCommand` (p. 3044), `activemq::commands::Response` (p. 3078), `activemq::commands::SessionInfo` (p. 3190), `activemq::commands::ShutdownInfo` (p. 3251), `activemq::commands::TransactionInfo` (p. 3596), `activemq::commands::WireFormatInfo` (p. 3721), `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 383), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 383), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 383), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 383), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 383), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 383).

References `activemq::commands::BaseDataStructure::equals()`.

6.125.2.3 virtual int activemq::commands::BaseCommand::getCommandId () const [inline, virtual]

Gets the **Command** (p. 1107) Id of this **Message** (p. 2358).

Returns

Command (p. 1107) Id

Implements `activemq::commands::Command` (p. 1108).

Referenced by `copyDataStructure()`.

6.125.2.4 `virtual bool activemq::commands::BaseCommand::isBrokerInfo ()`
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1109).

Reimplemented in `activemq::commands::BrokerInfo` (p. 828).

6.125.2.5 `virtual bool activemq::commands::BaseCommand::isConnectionInfo ()`
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1109).

Reimplemented in `activemq::commands::ConnectionInfo` (p. 1261).

6.125.2.6 `virtual bool activemq::commands::BaseCommand::isConsumerInfo ()`
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1109).

Reimplemented in `activemq::commands::ConsumerInfo` (p. 1362).

6.125.2.7 `virtual bool activemq::commands::BaseCommand::isKeepAliveInfo ()`
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1109).

Reimplemented in `activemq::commands::KeepAliveInfo` (p. 2124).

6.125.2.8 `virtual bool activemq::commands::BaseCommand::isMessage () const`
`[inline, virtual]`

Implements `activemq::commands::Command` (p. 1109).

Reimplemented in `activemq::commands::Message` (p. 2369).

6.125.2.9 `virtual bool activemq::commands::BaseCommand::isMessageAck ()`
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1109).

Reimplemented in `activemq::commands::MessageAck` (p. 2397).

6.125.2.10 `virtual bool activemq::commands::BaseCommand::isMessageDispatch () const`
`[inline, virtual]`

Implements `activemq::commands::Command` (p. 1109).

Reimplemented in `activemq::commands::MessageDispatch` (p. 2429).

6.125.2.11 `virtual bool activemq::commands::BaseCommand::isMessageDispatchNotification () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1109).

Reimplemented in `activemq::commands::MessageDispatchNotification` (p. 2465).

6.125.2.12 `virtual bool activemq::commands::BaseCommand::isProducerAck () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1110).

Reimplemented in `activemq::commands::ProducerAck` (p. 2841).

6.125.2.13 `virtual bool activemq::commands::BaseCommand::isProducerInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1110).

Reimplemented in `activemq::commands::ProducerInfo` (p. 2900).

6.125.2.14 `virtual bool activemq::commands::BaseCommand::isRemoveInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1110).

Reimplemented in `activemq::commands::RemoveInfo` (p. 2990).

6.125.2.15 `virtual bool activemq::commands::BaseCommand::isRemoveSubscriptionInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1110).

Reimplemented in `activemq::commands::RemoveSubscriptionInfo` (p. 3018).

6.125.2.16 `virtual bool activemq::commands::BaseCommand::isResponse () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1110).

Reimplemented in `activemq::commands::Response` (p. 3078).

6.125.2.17 `virtual bool activemq::commands::BaseCommand::isResponseRequired () const [inline, virtual]`

Is a **Response** (p. 3076) required for this **Command** (p. 1107).

Returns

true if a response is required.

Implements **activemq::commands::Command** (p. 1110).

Referenced by `copyDataStructure()`.

6.125.2.18 `virtual bool activemq::commands::BaseCommand::isShutdownInfo () const [inline, virtual]`

Implements **activemq::commands::Command** (p. 1110).

Reimplemented in **activemq::commands::ShutdownInfo** (p. 3251).

6.125.2.19 `virtual bool activemq::commands::BaseCommand::isTransactionInfo () const [inline, virtual]`

Implements **activemq::commands::Command** (p. 1111).

Reimplemented in **activemq::commands::TransactionInfo** (p. 3597).

6.125.2.20 `virtual bool activemq::commands::BaseCommand::isWireFormatInfo () const [inline, virtual]`

Implements **activemq::commands::Command** (p. 1111).

Reimplemented in **activemq::commands::WireFormatInfo** (p. 3724).

6.125.2.21 `virtual void activemq::commands::BaseCommand::setCommandId (int id) [inline, virtual]`

Sets the **Command** (p. 1107) Id of this **Message** (p. 2358).

Parameters

id **Command** (p. 1107) Id

Implements **activemq::commands::Command** (p. 1111).

6.125.2.22 `virtual void activemq::commands::BaseCommand::setResponseRequired (const bool required) [inline, virtual]`

Set if this **Message** (p. 2358) requires a **Response** (p. 3076).

Parameters

required true if response is required

Implements **activemq::commands::Command** (p. 1111).

6.125.2.23 `virtual std::string activemq::commands::BaseCommand::toString () const [inline, virtual]`

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implements `activemq::commands::Command` (p. 1111).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 170), `activemq::commands::ActiveMQBytesMessage` (p. 205), `activemq::commands::ActiveMQMapMessage` (p. 328), `activemq::commands::ActiveMQMessage` (p. 355), `activemq::commands::ActiveMQObjectMessage` (p. 398), `activemq::commands::ActiveMQStreamMessage` (p. 495), `activemq::commands::ActiveMQTextMessage` (p. 609), `activemq::commands::BrokerInfo` (p. 829), `activemq::commands::ConnectionControl` (p. 1175), `activemq::commands::ConnectionError` (p. 1203), `activemq::commands::ConnectionInfo` (p. 1262), `activemq::commands::ConsumerControl` (p. 1305), `activemq::commands::ConsumerInfo` (p. 1363), `activemq::commands::ControlCommand` (p. 1392), `activemq::commands::DataArrayResponse` (p. 1425), `activemq::commands::DataResponse` (p. 1479), `activemq::commands::DestinationInfo` (p. 1617), `activemq::commands::ExceptionResponse` (p. 1722), `activemq::commands::FlushCommand` (p. 1814), `activemq::commands::IntegerResponse` (p. 1958), `activemq::commands::KeepAliveInfo` (p. 2124), `activemq::commands::Message` (p. 2372), `activemq::commands::MessageAck` (p. 2398), `activemq::commands::MessageDispatch` (p. 2430), `activemq::commands::MessageDispatchNotification` (p. 2465), `activemq::commands::MessagePull` (p. 2567), `activemq::commands::ProducerAck` (p. 2842), `activemq::commands::ProducerInfo` (p. 2901), `activemq::commands::RemoveInfo` (p. 2990), `activemq::commands::RemoveSubscriptionInfo` (p. 3018), `activemq::commands::ReplayCommand` (p. 3045), `activemq::commands::Response` (p. 3079), `activemq::commands::SessionInfo` (p. 3191), `activemq::commands::ShutdownInfo` (p. 3251), `activemq::commands::TransactionInfo` (p. 3597), and `activemq::commands::WireFormatInfo` (p. 3727).

References `activemq::commands::BaseDataStructure::toString()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseCommand.h`

6.126 activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for `BaseCommandMarshaller` (p. 701).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`:

Public Member Functions

- `BaseCommandMarshaller ()`
- `virtual ~BaseCommandMarshaller ()`

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.126.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 701). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.126.2 Constructor & Destructor Documentation

6.126.2.1 **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::BaseCommandMarshaller** () [inline]

6.126.2.2 **virtual activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::~~BaseCommandMarshaller** () [inline, virtual]

6.126.3 Member Function Documentation

6.126.3.1 **virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseMarshal** (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 172), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 211), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 330), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 501), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 611), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 833), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1178), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1210), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1269), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1312), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1371), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1399), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1431), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1493), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1624), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1732), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1824), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1968), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2135), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2409), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2444), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2476), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2578), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2852), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 2920), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3001), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3025), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3060), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3100), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3205), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3270), and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3608).

6.126.3.2 virtual void `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 173), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 212), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 502), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 611), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 834), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1179), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1210), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1270), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1313), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1372), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1399), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1432), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1494), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1624), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1733), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1825), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1969), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2135), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2409), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2444), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2477), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2529), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2578), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2853), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 2920), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3001), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3025), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3060), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3100), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3206), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3270), and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3609).

```
6.126.3.3 virtual int ac-
    tivemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal1
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, utils::BooleanStream * bs ) throw (
    decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 173), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 212), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 502), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 612), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 834), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1179), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1210), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1270), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1313), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1372), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1399), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1432), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1494), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1624), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1733), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1825), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1969), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2135), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2409), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2444), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2477), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2530), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2578), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2853), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 2920), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3001), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3025), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3060), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3101), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3206), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3270), and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3609).

```

6.126.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 174), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 213), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 358), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 402), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 502), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 612), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 834), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1180), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1211), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1270), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1313), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1372), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1400), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1433), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1495), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1625), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1734), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1825), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1970), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2136), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2410), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2445), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2477), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2531), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2579), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2853), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 2921), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3002), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3026), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3061), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3101),

activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3206), activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3271), and activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3609).

6.126.3.5 virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements activemq::wireformat::openwire::marshal::DataStreamMarshaller (p. 1546).

Reimplemented in activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 174), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (p. 213), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (p. 332), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (p. 358), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (p. 402), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (p. 503), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (p. 612), activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 835), activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1180), activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1211), activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1271), activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1314), activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1373), activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1400), activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1433), activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1495), activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1625), activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1734), activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1826), activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1970), activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2136), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2410), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2445), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (p. 2478), activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2531), activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2579), activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2854), activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2921),

activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3002), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 3026), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 3061), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3102), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 3207), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 3271), and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 3610).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h`

6.127 **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 708).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.127.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 708). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.127.2 Constructor & Destructor Documentation

6.127.2.1 `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::BaseCommandMarshaller ()` [inline]

6.127.2.2 `virtual activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::~~BaseCommandMarshaller ()` [inline, virtual]

6.127.3 Member Function Documentation

6.127.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 180), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 219), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 338), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 365), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 409), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 509), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 615), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 837), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1182), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1214), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1273), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1316), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1375), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1403), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1435), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1497),

activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1628),
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1740),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1828),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1972),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2139),
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2413),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2452),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller
 (p. 2480), activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2537),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2582),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2848),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2904),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3013),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller
 (p. 3041), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller
 (p. 3048), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3086),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3213),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3274), and
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3616).

6.127.3.2 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::looseUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**
 (p. 181), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**
 (p. 220), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**
 (p. 339), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**
 (p. 365), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**
 (p. 409), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**
 (p. 510), **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**
 (p. 615), **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller** (p. 838),
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1183),
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1214),
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1274),
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1317),
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1376),
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1403),
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1436),

activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1498),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1628),
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1741),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1829),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1973),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2139),
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2413),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2452),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller
 (p. 2481), activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2537),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2582),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2849),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2904),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3013),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller
 (p. 3041), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller
 (p. 3048), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3087),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3214),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3274), and
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3617).

6.127.3.3 virtual int `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal1`
 (`OpenWireFormat * wireFormat`, `commands::DataStructure`
 * `dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`
 (p. 181), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`
 (p. 220), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`
 (p. 339), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`
 (p. 365), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`
 (p. 409), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`
 (p. 510), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`
 (p. 615), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 838),
`activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1183),

activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1214),
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1274),
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1317),
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1376),
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1403),
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1436),
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1498),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1628),
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1741),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1829),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1973),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2139),
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2413),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2452),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller
 (p. 2481), activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2538),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2582),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2849),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2904),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3013),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller
 (p. 3041), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller
 (p. 3048), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3087),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3214),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3274), and
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3617).

6.127.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal2
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataOutputStream * *dataOut*,**
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**
 (p. 181), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**
 (p. 220), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**
 (p. 339), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**
 (p. 366), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**

(p. 410), [activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller](#) (p. 510), [activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller](#) (p. 616), [activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller](#) (p. 838), [activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller](#) (p. 1183), [activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller](#) (p. 1215), [activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller](#) (p. 1274), [activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller](#) (p. 1317), [activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller](#) (p. 1376), [activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller](#) (p. 1404), [activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller](#) (p. 1437), [activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller](#) (p. 1499), [activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller](#) (p. 1629), [activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller](#) (p. 1742), [activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller](#) (p. 1829), [activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller](#) (p. 1974), [activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller](#) (p. 2140), [activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller](#) (p. 2414), [activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller](#) (p. 2453), [activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller](#) (p. 2481), [activemq::wireformat::openwire::marshal::v4::MessageMarshaller](#) (p. 2539), [activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller](#) (p. 2583), [activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller](#) (p. 2849), [activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller](#) (p. 2905), [activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller](#) (p. 3014), [activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller](#) (p. 3042), [activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller](#) (p. 3049), [activemq::wireformat::openwire::marshal::v4::ResponseMarshaller](#) (p. 3088), [activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller](#) (p. 3214), [activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller](#) (p. 3275), and [activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller](#) (p. 3617).

6.127.3.5 virtual void `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements [activemq::wireformat::openwire::marshal::DataStreamMarshaller](#) (p. 1546).

Reimplemented in [activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller](#) (p. 182), [activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller](#)

(p. 221), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 340), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 366), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 410), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 511), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 616), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 839), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1184), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1215), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1275), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1318), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1377), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1404), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1437), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1499), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1629), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1742), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1830), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1974), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2140), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2414), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2453), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2482), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2583), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2850), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2905), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3014), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3042), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3049), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3088), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3215), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3275), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3618).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h`

6.128 `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `BaseCommandMarshaller` (p. 714).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`:

Public Member Functions

- `BaseCommandMarshaller ()`
- `virtual ~BaseCommandMarshaller ()`

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.128.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 714). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.128.2 Constructor & Destructor Documentation

6.128.2.1 activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::BaseCommandMarshaller () [inline]

6.128.2.2 virtual activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::~~BaseCommandMarshaller () [inline, virtual]

6.128.3 Member Function Documentation

6.128.3.1 virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 176), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 215), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 334), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 361), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 405), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 505), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 619), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 841), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1186), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1218), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1277), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1320), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1379), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1407), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1439), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1501), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1632), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1744), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1832), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1976), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2147), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2417), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2456), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2484), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2586), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2864), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2912), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3005), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3021), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3052), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3104), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3201), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3262), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3604).

6.128.3.2 virtual void ac-

```

activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 177), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 216), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 335), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 361), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 405), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 506), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 619), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 842), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1187), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1218), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1278), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1321), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1380), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1407), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1440), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1632), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1745), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1833), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1977), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2147), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2417), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2456), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2485), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2541), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2586), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2865), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2912), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3005), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3021), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3052), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3105), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3202), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3262), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3605).

6.128.3.3 virtual int `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal1`
(`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 177), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 216), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 335), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 361), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 405), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 506), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 619), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 842), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1187), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1218), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1278), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1321), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1380), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1407), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1440), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1632), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1745), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1833), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1977), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2147), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2417), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2456), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2485), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2542), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2586), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2865), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2912), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3005), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3022), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3052), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3105), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3202), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3262), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3605).

6.128.3.4 virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 177), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 216), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 335), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 362), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 406), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 506), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 620), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 842), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1187), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1219), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1278), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1321), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1380), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1408), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1441), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1503), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1633), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1746), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1833), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1978), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2148), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2418), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2457), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2485), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2587), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2865), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2913), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3006), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3022), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3053), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3106),

`activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3202), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3263), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3605).

6.128.3.5 `virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 178), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 217), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 336), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 362), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 406), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 507), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 620), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 843), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1188), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1219), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1279), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1322), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1381), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1408), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1441), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1503), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1633), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1746), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1834), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1978), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2148), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2418), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2457), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2486), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2587), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2866), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2913),

`activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3006), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3022), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3053), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3106), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3203), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3263), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3606).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h`

6.129 `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `BaseCommandMarshaller` (p. 721).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`:

Public Member Functions

- `BaseCommandMarshaller` ()
- virtual `~BaseCommandMarshaller` ()
- virtual void `tightUnmarshal` (`OpenWireFormat` *wireFormat, `commands::DataStructure` *dataStructure, `decaf::io::DataInputStream` *dataIn, `utils::BooleanStream` *bs) throw (`decaf::io::IOException`)
Un-marshal an object instance from the data input stream.
- virtual int `tightMarshal1` (`OpenWireFormat` *wireFormat, `commands::DataStructure` *dataStructure, `utils::BooleanStream` *bs) throw (`decaf::io::IOException`)
Write the booleans that this object uses to a BooleanStream.
- virtual void `tightMarshal2` (`OpenWireFormat` *wireFormat, `commands::DataStructure` *dataStructure, `decaf::io::DataOutputStream` *dataOut, `utils::BooleanStream` *bs) throw (`decaf::io::IOException`)
Write a object instance to data output stream.
- virtual void `looseUnmarshal` (`OpenWireFormat` *wireFormat, `commands::DataStructure` *dataStructure, `decaf::io::DataInputStream` *dataIn) throw (`decaf::io::IOException`)
Un-marshal an object instance from the data input stream.
- virtual void `looseMarshal` (`OpenWireFormat` *wireFormat, `commands::DataStructure` *dataStructure, `decaf::io::DataOutputStream` *dataOut) throw (`decaf::io::IOException`)
Write a object instance to data output stream.

6.129.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 721). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.129.2 Constructor & Destructor Documentation

6.129.2.1 `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::BaseCommandMarshaller()` [inline]

6.129.2.2 `virtual activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::~~BaseCommandMarshaller()` [inline, virtual]

6.129.3 Member Function Documentation

6.129.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 223), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 342), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 369), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 413), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 513), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 623), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 845), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1190), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1222), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1281), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1324), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1383), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1411), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1443), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1481),

activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1640), activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1736), activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1836), activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1980), activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2143), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2421), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2448), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller (p. 2488), activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2524), activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2574), activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2856), activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2916), activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3009), activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller (p. 3037), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller (p. 3068), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3095), activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3197), activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3266), and activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3600).

6.129.3.2 virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 189), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 224), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 343), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 369), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 413), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 514), **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 623), **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller** (p. 846), **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller** (p. 1191), **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller** (p. 1222), **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller** (p. 1282), **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller** (p. 1325), **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller** (p. 1384), **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller** (p. 1411), **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1444),

activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1482),
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1640), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1737),
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1837),
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1981),
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2143),
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2421), **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller** (p. 2448), **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller** (p. 2489), **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2525),
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2574),
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2857),
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2916),
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3009), **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller** (p. 3037), **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller** (p. 3068), **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3096),
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3198), **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** (p. 3266), and **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller** (p. 3601).

6.129.3.3 virtual int activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal1
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - **BooleanStream** stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 189), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 224), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 343), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 369), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 413), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 514), **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 623), **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller** (p. 846), **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller** (p. 1191),

activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1222),
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1282),
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1325),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1384),
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1411),
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1444),
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1482),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1640),
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1737),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1837),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1981),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2143),
 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2421),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2448),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2489),
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2526),
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2574),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2857),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2916),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3009),
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
 (p. 3037),
 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
 (p. 3068),
 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3096),
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3198),
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3266), and
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3601).

6.129.3.4 virtual void `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal2`
 (`OpenWireFormat * wireFormat`, `commands::DataStructure`
 * `dataStructure`, `decaf::io::DataOutputStream * dataOut`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`
 (p. 189), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`
 (p. 224), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`
 (p. 343), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`
 (p. 370), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`

(p. 414), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 514), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 624), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 846), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1191), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1223), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1282), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1325), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1384), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1412), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1445), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1483), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1641), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1738), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1837), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1982), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2144), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2422), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2449), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2489), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2526), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2575), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2857), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2917), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3010), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 3038), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 3069), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3097), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3198), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3267), and `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3601).

6.129.3.5 `virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 190), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`

(p. 225), [activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller](#) (p. 344), [activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller](#) (p. 370), [activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller](#) (p. 414), [activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller](#) (p. 515), [activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller](#) (p. 624), [activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller](#) (p. 847), [activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller](#) (p. 1192), [activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller](#) (p. 1223), [activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller](#) (p. 1283), [activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller](#) (p. 1326), [activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller](#) (p. 1385), [activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller](#) (p. 1412), [activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller](#) (p. 1445), [activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller](#) (p. 1483), [activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller](#) (p. 1641), [activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller](#) (p. 1738), [activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller](#) (p. 1838), [activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller](#) (p. 1982), [activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller](#) (p. 2144), [activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller](#) (p. 2422), [activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller](#) (p. 2449), [activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller](#) (p. 2490), [activemq::wireformat::openwire::marshal::v5::MessageMarshaller](#) (p. 2527), [activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller](#) (p. 2575), [activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller](#) (p. 2858), [activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller](#) (p. 2917), [activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller](#) (p. 3010), [activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller](#) (p. 3038), [activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller](#) (p. 3069), [activemq::wireformat::openwire::marshal::v5::ResponseMarshaller](#) (p. 3097), [activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller](#) (p. 3199), [activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller](#) (p. 3267), and [activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller](#) (p. 3602).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h`

6.130 activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 728).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller`:

Public Member Functions

- `BaseCommandMarshaller ()`
- `virtual ~BaseCommandMarshaller ()`

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.130.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 728). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.130.2 Constructor & Destructor Documentation

6.130.2.1 **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::BaseCommandMarshaller** () [inline]

6.130.2.2 **virtual activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::~~BaseCommandMarshaller** () [inline, virtual]

6.130.3 Member Function Documentation

6.130.3.1 **virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::looseMarshal** (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 227), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 350), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 421), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 521), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 627), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 849), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1194), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1226), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1285), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1328), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1387), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1415), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1447), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1485), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1636), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1724), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1816), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1960), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2127), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2401), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2460), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2468), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2590), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 2860), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 2924), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 2997), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3033), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3064), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3193), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3254), and `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3612).

6.130.3.2 virtual void `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 193), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 228), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 351), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 421), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 522), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 627), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 850), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1195), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1226), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1286), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1329), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1388), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1415), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1448), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1486), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1636), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1725), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1817), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1961), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2127), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2401), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2460), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2469), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2590), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 2861), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 2924), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 2997), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3033), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3064), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3194), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3254), and `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3613).

```
6.130.3.3 virtual int ac-
    tivemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightMarshal1
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, utils::BooleanStream * bs ) throw (
    decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 193), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 228), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 351), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 421), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 522), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 627), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 850), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1195), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1226), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1286), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1329), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1388), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1415), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1448), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1486), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1636), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1725), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1817), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1961), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2128), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2402), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2460), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2469), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2546), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2590), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 2861), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 2924), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 2997), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3033), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3064), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3110), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3194), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3255), and `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3613).

```

6.130.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 193), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 228), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 351), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 378), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 422), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 522), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 628), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 850), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1195), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1227), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1286), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1329), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1388), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1416), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1449), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1487), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1637), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1726), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1817), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1962), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2128), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2402), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2461), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2469), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2547), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2591), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 2861), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 2925), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 2998), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3034), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3065), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3110),

activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3194), activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3255), and activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3613).

6.130.3.5 virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements activemq::wireformat::openwire::marshal::DataStreamMarshaller (p. 1546).

Reimplemented in activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller (p. 194), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller (p. 229), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller (p. 352), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller (p. 378), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller (p. 422), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller (p. 523), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller (p. 628), activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 851), activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1196), activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1227), activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1287), activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1330), activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1389), activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1416), activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1449), activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1487), activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1637), activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1726), activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1818), activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 1962), activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2128), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller (p. 2402), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller (p. 2461), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller (p. 2470), activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2547), activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2591), activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 2862), activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 2925),

activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 2998), **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller** (p. 3034), **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller** (p. 3065), **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3111), **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller** (p. 3195), **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller** (p. 3255), and **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller** (p. 3614).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h`

6.131 **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 734).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.131.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 734). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.131.2 Constructor & Destructor Documentation

6.131.2.1 `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::BaseCommandMarshaller()` [inline]

6.131.2.2 `virtual activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::~~BaseCommandMarshaller()` [inline, virtual]

6.131.3 Member Function Documentation

6.131.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 346), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 373), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 417), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 517), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 631), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 853), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1198), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1206), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1265), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1308), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1367), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1395), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1427), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1489),

activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1620),
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1728),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1820),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1964),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2131),
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2405),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2440),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2472), activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2533),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2570),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2844),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2908),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2993),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 3029), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 3056), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3091),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3209),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3258), and
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3620).

6.131.3.2 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**
 (p. 185), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**
 (p. 232), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**
 (p. 347), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller**
 (p. 373), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller**
 (p. 417), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**
 (p. 518), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**
 (p. 631), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 854),
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1199),
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1206),
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1266),
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1309),
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1368),
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1395),
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1428),

activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1490),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1620),
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1729),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1821),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1965),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2131),
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2405),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2440),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2473), activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2533),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2570),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2845),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2908),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2993),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 3029), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 3056), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3091),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3210),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3258), and
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3621).

6.131.3.3 virtual int activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal1
 (*OpenWireFormat* * *wireFormat*, *commands::DataStructure*
 * *dataStructure*, *utils::BooleanStream* * *bs*) throw (*decaf::io::IOException*) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements *activemq::wireformat::openwire::marshal::DataStreamMarshaller* (p. 1532).

Reimplemented in *activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller*
 (p. 185), *activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller*
 (p. 232), *activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller*
 (p. 347), *activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller*
 (p. 373), *activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller*
 (p. 417), *activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller*
 (p. 518), *activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller*
 (p. 631), *activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller* (p. 854),
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1199),

activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1207),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1266),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1309),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1368),
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1396),
 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1428),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1490),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1621),
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1729),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1821),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1965),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2131),
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2405),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2440),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2473), activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2534),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2571),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2845),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2908),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2994),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 3029), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 3056), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3092),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3210),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3258), and
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3621).

6.131.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal2
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataOutputStream * *dataOut*,**
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**
 (p. 185), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**
 (p. 232), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**
 (p. 347), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller**
 (p. 374), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller**

(p. 418), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 518), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 632), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 854), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1199), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1207), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1266), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1309), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1369), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1396), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1429), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1491), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1621), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1730), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1821), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1966), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2132), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2406), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2441), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2473), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2571), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2845), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 2909), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2994), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 3030), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3057), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3092), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3210), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3259), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3621).

6.131.3.5 `virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 186), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller`

(p. 233), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 348), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 374), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 418), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 519), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 632), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 855), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1200), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1207), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1267), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1310), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1369), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1396), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1429), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1491), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1621), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1730), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1822), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1966), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2132), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2406), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2441), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2474), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2571), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2846), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 2909), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2994), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 3030), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3057), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3093), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3211), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3259), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h`

6.132 `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller` Class Reference

Base class for all Marshallers that marshal DataStructures to and from the wire using the Open-Wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

Inherits `activemq::wireformat::openwire::marshal::DataStreamMarshaller`.

Inherited by `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller`, `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`, `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller`, `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller`, `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller`, and `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller`.

tivemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v1::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller,	ac-

tivemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller,	and ac-
tivemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller.	

Public Member Functions

- virtual `~BaseDataStreamMarshaller()`
- virtual `int tightMarshal1(OpenWireFormat *format, AMQCPP_UNUSED, com-`

mands::DataStructure *command AMQCPP_UNUSED, **utils::BooleanStream** *bs AMQCPP_UNUSED) throw (`decaf::io::IOException`)

Tight Marshal to the given stream.

- virtual void **tightMarshal2** (`OpenWireFormat` *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataOutputStream** *ds AMQCPP_UNUSED, **utils::BooleanStream** *bs AMQCPP_UNUSED) throw (`decaf::io::IOException`)

Tight Marshal to the given stream.

- virtual void **tightUnmarshal** (`OpenWireFormat` *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataInputStream** *dis AMQCPP_UNUSED, **utils::BooleanStream** *bs AMQCPP_UNUSED) throw (`decaf::io::IOException`)

Tight Un-Marshal to the given stream.

- virtual void **looseMarshal** (`OpenWireFormat` *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataOutputStream** *ds AMQCPP_UNUSED) throw (`decaf::io::IOException`)

Tight Marshal to the given stream.

- virtual void **looseUnmarshal** (`OpenWireFormat` *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataInputStream** *dis AMQCPP_UNUSED) throw (`decaf::io::IOException`)

Loose Un-Marshal to the given stream.

Static Public Member Functions

- static std::string **toString** (const **commands::MessageId** *id)
Converts the object to a String.
- static std::string **toString** (const **commands::ProducerId** *id)
Converts the object to a String.
- static std::string **toString** (const **commands::TransactionId** *txnId)
Converts the given transaction ID into a String.
- static std::string **toHexFromBytes** (const std::vector< unsigned char > &data)
given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

Protected Member Functions

- virtual **commands::DataStructure** * **tightUnmarshalCachedObject** (`OpenWireFormat` *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (`decaf::io::IOException`)
Tight Unmarshal the cached object.

- virtual int **tightMarshalCachedObject1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.
- virtual void **tightMarshalCachedObject2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tightly marshals the passed DataStructure based object to the passed streams returning nothing.
- virtual void **looseMarshalCachedObject** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Loosely marshals the passed DataStructure based object to the passed stream returning nothing.
- virtual **commands::DataStructure** * **looseUnmarshalCachedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Loose Unmarshal the cached object.
- virtual int **tightMarshalNestedObject1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.
- virtual void **tightMarshalNestedObject2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tightly marshals the passed DataStructure based object to the passed streams returning nothing.
- virtual **commands::DataStructure** * **tightUnmarshalNestedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tight Unmarshal the nested object.
- virtual **commands::DataStructure** * **looseUnmarshalNestedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Loose Unmarshal the nested object.
- virtual void **looseMarshalNestedObject** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Loose marshal the nested object.
- virtual std::string **tightUnmarshalString** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Performs Tight Unmarshaling of String Objects.

- virtual int **tightMarshalString1** (const std::string &value, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.
- virtual void **tightMarshalString2** (const std::string &value, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tight Marshals the passed string to the streams passed.
- virtual void **looseMarshalString** (const std::string value, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Loose Marshal the String to the DataOuputStream passed.
- virtual std::string **looseUnmarshalString** (**decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Loose Un-Marshall the String to the DataOuputStream passed.
- virtual int **tightMarshalLong1** (**OpenWireFormat** *wireFormat, long long value, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tightly marshal the long long to the BooleanStream passed.
- virtual void **tightMarshalLong2** (**OpenWireFormat** *wireFormat, long long value, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tightly marshal the long long to the Streams passed.
- virtual long long **tightUnmarshalLong** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tight marshal the long long type.
- virtual void **looseMarshalLong** (**OpenWireFormat** *wireFormat, long long value, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Tightly marshal the long long to the BooleanStream passed.
- virtual long long **looseUnmarshalLong** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Loose marshal the long long type.
- virtual std::vector< unsigned char > **tightUnmarshalByteArray** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tight Unmarshal an array of char.
- virtual std::vector< unsigned char > **looseUnmarshalByteArray** (**decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Loose Unmarshal an array of char.
- virtual std::vector< unsigned char > **tightUnmarshalConstByteArray** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs, int size) throw (**decaf::io::IOException**)

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

- virtual `std::vector< unsigned char > looseUnmarshalConstByteArray (decaf::io::DataInputStream *dataIn, int size) throw (decaf::io::IOException)`

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

- virtual `commands::DataStructure * tightUnmarshalBrokerError (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Tight Unmarshal the Error object.

- virtual `int tightMarshalBrokerError1 (OpenWireFormat *wireFormat, commands::DataStructure *data, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Tight Marshal the Error object.

- virtual `void tightMarshalBrokerError2 (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Tight Marshal the Error object.

- virtual `commands::DataStructure * looseUnmarshalBrokerError (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Loose Unmarshal the Error object.

- virtual `void looseMarshalBrokerError (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Tight Marshal the Error object.

- `template<typename T > int tightMarshalObjectArray1 (OpenWireFormat *wireFormat, std::vector< T > objects, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.

- `template<typename T > void tightMarshalObjectArray2 (OpenWireFormat *wireFormat, std::vector< T > objects, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

- `template<typename T > void looseMarshalObjectArray (OpenWireFormat *wireFormat, std::vector< T > objects, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

- virtual std::string readAsciiString (decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Given an DataInputStream read a know ASCII formatted string from the input and return that string.

6.132.1 Detailed Description

Base class for all Marshallers that marshal DataStructures to and from the wire using the Open-Wire protocol.

Since

2.0

6.132.2 Constructor & Destructor Documentation

- 6.132.2.1** virtual
 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::~BaseDataStreamMarshaller () [inline, virtual]

6.132.3 Member Function Documentation

- 6.132.3.1** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshal (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, decaf::io::DataOutputStream *ds *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Tight Marshal to the given stream.

Parameters

format - The OpenwireFormat properties

command - the object to Marshal

ds - DataOutputStream to marshal to

Exceptions

IOException if an error occurs.

- 6.132.3.2** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalBrokerError (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [protected, virtual]

Tight Marshal the Error object.

Parameters

wireFormat - The OpenwireFormat properties

data - Error to Marshal

dataOut - stream to write marshalled data to

Exceptions

IOException if an error occurs.

6.132.3.3 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalCachedObject (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [protected, virtual]

Loosely marshals the passed DataStructure based object to the passed stream returning nothing.

Parameters

wireFormat - The OpenWireFormat properties

data - DataStructure Object Pointer to marshal

dataOut - stream to write marshaled data to

Exceptions

IOException if an error occurs.

6.132.3.4 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalLong (OpenWireFormat * *wireFormat*, long long *value*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [protected, virtual]

Tightly marshal the long long to the BooleanStream passed.

Parameters

wireFormat - The OpenWireFormat properties

value - long long to marshal

dataOut - DataOutputStream to marshal to.

Exceptions

IOException if an error occurs.

6.132.3.5 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalNestedObject (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [protected, virtual]

Loose marshal the nested object.

Parameters

wireFormat - The OpenwireFormat properties
object - DataStructure Object Pointer to marshal
dataOut - stream to write marshaled data to

Exceptions

IOException if an error occurs.

```
6.132.3.6 template<typename T > void ac-
tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectA
( OpenWireFormat * wireFormat, std::vector< T > objects,
  decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException
) [inline, protected]
```

Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

Parameters

wireFormat - The OpenwireFormat properties
objects - array of DataStructure object pointers.
dataOut - stream to write marshalled data to

Returns

size of the marshalled data

Exceptions

IOException if an error occurs.

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, and AMQ_CATCHALL_THROW.

```
6.132.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalString
( const std::string value, decaf::io::DataOutputStream * dataOut )
throw ( decaf::io::IOException ) [protected, virtual]
```

Loose Marshal the String to the DataOuputStream passed.

Parameters

value - string to marshal
dataOut - stream to write marshaled form to

Exceptions

IOException if an error occurs.

6.132.3.8 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis AMQCPP_UNUSED) throw (decaf::io::IOException)` [inline, virtual]

Loose Un-Marshal to the given stream.

Parameters

format - The OpenwireFormat properties
command - the object to Un-Marshal
dis - the DataInputStream to Un-Marshal from

Exceptions

IOException if an error occurs.

6.132.3.9 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBroken (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Loose Unmarshal the Error object.

Parameters

wireFormat - The OpenwireFormat properties
dataIn - stream to read marshalled form from

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

6.132.3.10 `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBytes (decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Loose Unmarshal an array of char.

Parameters

dataIn - the DataInputStream to Un-Marshal from

Returns

the unmarshalled vector of chars.

Exceptions

IOException if an error occurs.

6.132.3.11 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalCached (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Loose Unmarshal the cached object.

Parameters

wireFormat - The OpenwireFormat properties
dataIn - stream to read marshaled form from

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

6.132.3.12 `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalConst (decaf::io::DataInputStream * dataIn, int size) throw (decaf::io::IOException)` [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

Parameters

dataIn - the DataInputStream to Un-Marshall from
size - size of the const array to unmarshal

Returns

the unmarshaled vector of chars.

Exceptions

IOException if an error occurs.

6.132.3.13 `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalLong (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Loose marshal the long long type.

Parameters

wireFormat - The OpenwireFormat properties
dataIn - stream to read marshaled form from

Returns

the unmarshaled long long

Exceptions

IOException if an error occurs.

6.132.3.14 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalNested (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Loose Unmarshal the nested object.

Parameters

wireFormat - The OpenWireFormat properties

dataIn - stream to read marshaled form from

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

6.132.3.15 `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalString (decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Loose Un-Marshal the String to the DataOutputStream passed.

Parameters

dataIn - stream to read marshaled form from

Returns

the unmarshaled string

Exceptions

IOException if an error occurs.

6.132.3.16 `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::readAsciiString (decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Given an DataInputStream read a know ASCII formatted string from the input and return that string.

Parameters

dataIn - DataInputStream to read from

Returns

string value read from stream

6.132.3.17 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal1 (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, utils::BooleanStream *bs *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Tight Marshal to the given stream.

Parameters

format - The OpenWireFormat properties

command - the object to Marshal

bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

6.132.3.18 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal2 (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, decaf::io::DataOutputStream *ds *AMQCPP_UNUSED*, utils::BooleanStream *bs *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Tight Marshal to the given stream.

Parameters

format - The OpenWireFormat properties

command - the object to Marshal

ds - the DataOutputStream to Marshal to

bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

6.132.3.19 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBroker(OpenWireFormat * wireFormat, commands::DataStructure * data, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Marshal the Error object.

Parameters

wireFormat - The OpenwireFormat properties

data - Error to Marshal

bs - boolean stream to marshal to.

Returns

size of the marshalled data

Exceptions

IOException if an error occurs.

6.132.3.20 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBroker(OpenWireFormat * wireFormat, commands::DataStructure * data, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Marshal the Error object.

Parameters

wireFormat - The OpenwireFormat properties

data - Error to Marshal

dataOut - stream to write marshalled data to

bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

6.132.3.21 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCached(OpenWireFormat * wireFormat, commands::DataStructure * data, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

Parameters

wireFormat - The OpenwireFormat properties

data - DataStructure Object Pointer to marshal

bs - boolean stream to marshal to.

Returns

size of data written.

Exceptions

IOException if an error occurs.

6.132.3.22 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCached
(OpenWireFormat * *wireFormat*, commands::DataStructure * *data*,
decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*)
throw (decaf::io::IOException) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

Parameters

wireFormat - The OpenWireFormat properties

data - DataStructure Object Pointer to marshal

bs - boolean stream to marshal to.

dataOut - stream to write marshaled data to

Exceptions

IOException if an error occurs.

6.132.3.23 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong1
(OpenWireFormat * *wireFormat*, long long *value*,
utils::BooleanStream * *bs*) throw (decaf::io::IOException)
[protected, virtual]

Tightly marshal the long long to the BooleanStream passed.

Parameters

wireFormat - The OpenWireFormat properties

value - long long to marshal

bs - boolean stream to marshal to.

Returns

size of data written.

Exceptions

IOException if an error occurs.

6.132.3.24 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong2 (OpenWireFormat * wireFormat, long long value, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tightly marshal the long long to the Streams passed.

Parameters

wireFormat - The OpenwireFormat properties
value - long long to marshal
dataOut - stream to write marshaled form to
bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

6.132.3.25 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNested (OpenWireFormat * wireFormat, commands::DataStructure * object, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

Parameters

wireFormat - The OpenwireFormat properties
object - DataStructure Object Pointer to marshal
bs - boolean stream to marshal to.

Returns

size of data written.

Exceptions

IOException if an error occurs.

6.132.3.26 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNested (OpenWireFormat * wireFormat, commands::DataStructure * object, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

Parameters

wireFormat - The OpenwireFormat properties

object - DataStructure Object Pointer to marshal

bs - boolean stream to marshal to.

dataOut - stream to write marshaled data to

Exceptions

IOException if an error occurs.

```
6.132.3.27  template<typename T > int ac-
             tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObject
             ( OpenWireFormat * wireFormat, std::vector< T > objects,
             utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [inline,
             protected]
```

Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.

Parameters

wireFormat - The OpenwireFormat properties

objects - array of DataStructure object pointers.

bs - boolean stream to marshal to.

Returns

size of the marshalled data

Exceptions

IOException if an error occurs.

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, and AMQ_CATCHALL_THROW.

```
6.132.3.28  template<typename T > void ac-
             tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObject
             ( OpenWireFormat * wireFormat, std::vector< T > objects,
             decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs )
             throw ( decaf::io::IOException ) [inline, protected]
```

Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

Parameters

wireFormat - The OpenwireFormat properties

objects - array of DataStructure object pointers.

dataOut - stream to write marshalled data to

bs - boolean stream to marshal to.

Returns

size of the marshalled data

Exceptions

IOException if an error occurs.

References `AMQ_CATCH_EXCEPTION_CONVERT`, `AMQ_CATCH_RETHROW`, and `AMQ_CATCHALL_THROW`.

6.132.3.29 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString1 (const std::string & value, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

Parameters

value - string to marshal
bs - BooleanStream to use.

Returns

size of marshaled string.

Exceptions

IOException if an error occurs.

6.132.3.30 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString2 (const std::string & value, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Marshals the passed string to the streams passed.

Parameters

value - string to marshal
dataOut - the DataOutputStream to Marshal to
bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

6.132.3.31 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw (decaf::io::IOException)` [inline, virtual]

Tight Un-Marshal to the given stream.

Parameters

format - The OpenwireFormat properties
command - the object to Un-Marshall
dis - the DataInputStream to Un-Marshall from
bs - boolean stream to Un-Marshall from.

Exceptions

IOException if an error occurs.

```
6.132.3.32 virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalBroken
( OpenWireFormat * wireFormat, decaf::io::DataInputStream *
dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[protected, virtual]
```

Tight Unmarshal the Error object.

Parameters

wireFormat - The OpenwireFormat properties
dataIn - stream to read marshalled form from
bs - boolean stream to marshal to.

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

```
6.132.3.33 virtual std::vector<unsigned char> ac-
tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalByte
( decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs )
throw ( decaf::io::IOException ) [protected, virtual]
```

Tight Unmarshal an array of char.

Parameters

dataIn - the DataInputStream to Un-Marshall from
bs - boolean stream to unmarshal from.

Returns

the unmarshaled vector of chars.

Exceptions

IOException if an error occurs.

6.132.3.34 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalCached (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Unmarshal the cached object.

Parameters

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

bs - boolean stream to marshal to.

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

6.132.3.35 `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalConst (decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs, int size) throw (decaf::io::IOException)` [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

Parameters

dataIn - the DataInputStream to Un-Marshall from

bs - boolean stream to unmarshal from.

size - size of the const array to unmarshal

Returns

the unmarshaled vector of chars.

Exceptions

IOException if an error occurs.

6.132.3.36 `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalLong (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight marshal the long long type.

Parameters

wireFormat - The OpenwireFormat properties
dataIn - stream to read marshaled form from
bs - boolean stream to marshal to.

Returns

the unmarshaled long long

Exceptions

IOException if an error occurs.

6.132.3.37 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalNested(OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Unmarshal the nested object.

Parameters

wireFormat - The OpenwireFormat properties
dataIn - stream to read marshaled form from
bs - boolean stream to marshal to.

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

6.132.3.38 `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalString(decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Performs Tight Unmarshaling of String Objects.

Parameters

dataIn - the DataInputStream to Un-Marshall from
bs - boolean stream to unmarshal from.

Returns

the unmarshaled string.

Exceptions

IOException if an error occurs.

6.132.3.39 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toHexFromBytes
(const std::vector< unsigned char > & data) [static]`

given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

Parameters

data - unsigned char data array pointer

Returns

a string coded in hex that represents the data

6.132.3.40 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString
(const commands::TransactionId * txnId) [static]`

Converts the given transaction ID into a String.

Parameters

txnId - TransactionId poitner

Returns

string representation of the id

6.132.3.41 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString
(const commands::ProducerId * id) [static]`

Converts the object to a String.

Parameters

id - ProducerId pointer

Returns

string representing the id

6.132.3.42 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString
(const commands::MessageId * id) [static]`

Converts the object to a String.

Parameters

id - MessageId pointer

Returns

string representing the id

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getCMSMessageID()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h`

6.133 activemq::commands::BaseDataStructure Class Reference

```
#include <src/main/activemq/commands/BaseDataStructure.h>
```

Inheritance diagram for `activemq::commands::BaseDataStructure`:

Public Member Functions

- virtual `~BaseDataStructure()`
- virtual `bool isMarshalAware() const`
Determine if this object is aware of marshaling and should have its before and after marshaling methods called.
- virtual `void beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)`
Perform any processing needed before an marshal.
- virtual `void afterMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)`
Perform any processing needed after an unmarshal.
- virtual `void beforeUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)`
Perform any processing needed before an unmarshal.
- virtual `void afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)`
Perform any processing needed after an unmarshal.
- virtual `void setMarshaledForm (wireformat::WireFormat *wireFormat AMQCPP_UNUSED, const std::vector< char > &data AMQCPP_UNUSED)`
Called to set the data to this object that will contain the objects marshaled form.
- virtual `std::vector< unsigned char > getMarshaledForm (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)`
Called to get the data to this object that will contain the objects marshaled form.

- virtual void **copyDataStructure** (const **DataStructure** *src AMQCPP_UNUSED)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value AMQCPP_UNUSED) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

6.133.1 Constructor & Destructor Documentation

- 6.133.1.1 virtual activemq::commands::BaseDataStructure::~~BaseDataStructure () [inline, virtual]

6.133.2 Member Function Documentation

- 6.133.2.1 virtual void activemq::commands::BaseDataStructure::afterMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException) [inline, virtual]

Perform any processing needed after an unmarshal.

Parameters

wireformat - the OpenWireFormat object in use.

- 6.133.2.2 virtual void activemq::commands::BaseDataStructure::afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException) [inline, virtual]

Perform any processing needed after an unmarshal.

Parameters

wireformat - the OpenWireFormat object in use.

Reimplemented in **activemq::commands::Message** (p. 2362), and **activemq::commands::WireFormatInfo** (p. 3720).

- 6.133.2.3 virtual void activemq::commands::BaseDataStructure::beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException) [inline, virtual]

Perform any processing needed before an marshal.

Parameters

wireformat - the OpenWireFormat object in use.

Reimplemented in **activemq::commands::Message** (p. 2362), and **activemq::commands::WireFormatInfo** (p. 3720).

6.133.2.4 `virtual void activemq::commands::BaseDataStructure::beforeUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException) [inline, virtual]`

Perform any processing needed before an unmarshal.

Parameters

wireformat - the OpenWireFormat object in use.

6.133.2.5 `virtual void activemq::commands::BaseDataStructure::copyDataStructure (const DataStructure *src AMQCPP_UNUSED) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented in `activemq::commands::BooleanExpression` (p. 787).

6.133.2.6 `virtual bool activemq::commands::BaseDataStructure::equals (const DataStructure *value AMQCPP_UNUSED) const [inline, virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Referenced by `activemq::commands::BooleanExpression::equals()`, and `activemq::commands::BaseCommand::equals()`.

6.133.2.7 `virtual std::vector<unsigned char> activemq::commands::BaseDataStructure::getMarshaledForm (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) [inline, virtual]`

Called to get the data to this object that will contain the objects marshaled form.

Parameters

wireFormat - the wireformat object to control unmarshaling

Returns

buffer that holds the objects data.

6.133.2.8 `virtual bool activemq::commands::BaseDataStructure::isMarshalAware () const [inline, virtual]`

Determine if this object is aware of marshaling and should have its before and after marshaling methods called.

Defaults to false.

Returns

true if aware of marshaling

Implements `activemq::wireformat::MarshalAware` (p. 2330).

Reimplemented in `activemq::commands::ActiveMQMapMessage` (p. 324), `activemq::commands::Message` (p. 2368), and `activemq::commands::WireFormatInfo` (p. 3723).

6.133.2.9 `virtual void activemq::commands::BaseDataStructure::setMarshaledForm (wireformat::WireFormat *wireFormat AMQCPP_UNUSED, const std::vector< char > &data AMQCPP_UNUSED) [inline, virtual]`

Called to set the data to this object that will contain the objects marshaled form.

Parameters

wireFormat - the wireformat object to control unmarshaling

data - vector of object binary data

6.133.2.10 `virtual std::string activemq::commands::BaseDataStructure::toString () const [inline, virtual]`

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implements `activemq::commands::DataStructure` (p. 1558).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 170), `activemq::commands::ActiveMQBytesMessage` (p. 205), `activemq::commands::ActiveMQDestination` (p. 288), `activemq::commands::ActiveMQMapMessage` (p. 328), `activemq::commands::ActiveMQMessage` (p. 355), `activemq::commands::ActiveMQObjectMessage` (p. 398), `activemq::commands::ActiveMQStreamMessage` (p. 495), `activemq::commands::ActiveMQTextMessage` (p. 609), `activemq::commands::BaseCommand` (p. 700), `activemq::commands::BooleanExpression` (p. 787), `activemq::commands::BrokerInfo` (p. 829), `activemq::commands::Command` (p. 1111), `activemq::commands::ConnectionControl` (p. 1175), `activemq::commands::ConnectionError` (p. 1203), `activemq::commands::ConnectionInfo` (p. 1262), `activemq::commands::ConsumerControl` (p. 1305), `activemq::commands::ConsumerInfo` (p. 1363), `activemq::commands::ControlCommand`

(p. 1392), `activemq::commands::DataArrayResponse` (p. 1425), `activemq::commands::DataResponse` (p. 1479), `activemq::commands::DestinationInfo` (p. 1617), `activemq::commands::DiscoveryEvent` (p. 1646), `activemq::commands::ExceptionResponse` (p. 1722), `activemq::commands::FlushCommand` (p. 1814), `activemq::commands::IntegerResponse` (p. 1958), `activemq::commands::JournalQueueAck` (p. 2016), `activemq::commands::JournalTopicAck` (p. 2044), `activemq::commands::JournalTrace` (p. 2071), `activemq::commands::JournalTransaction` (p. 2098), `activemq::commands::KeepAliveInfo` (p. 2124), `activemq::commands::LastPartialCommand` (p. 2158), `activemq::commands::Message` (p. 2372), `activemq::commands::MessageAck` (p. 2398), `activemq::commands::MessageDispatch` (p. 2430), `activemq::commands::MessageDispatchNotification` (p. 2465), `activemq::commands::MessagePull` (p. 2567), `activemq::commands::NetworkBridgeFilter` (p. 2616), `activemq::commands::PartialCommand` (p. 2730), `activemq::commands::ProducerAck` (p. 2842), `activemq::commands::ProducerInfo` (p. 2901), `activemq::commands::RemoveInfo` (p. 2990), `activemq::commands::RemoveSubscriptionInfo` (p. 3018), `activemq::commands::ReplayCommand` (p. 3045), `activemq::commands::Response` (p. 3079), `activemq::commands::SessionInfo` (p. 3191), `activemq::commands::ShutdownInfo` (p. 3251), `activemq::commands::SubscriptionInfo` (p. 3437), `activemq::commands::TransactionInfo` (p. 3597), and `activemq::commands::WireFormatInfo` (p. 3727).

Referenced by `activemq::commands::BooleanExpression::toString()`, and `activemq::commands::BaseCommand::toString()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseDataStructure.h`

6.134 `binary_function` Class Reference

Inheritance diagram for `binary_function`:

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

6.135 `decaf::net::BindException` Class Reference

```
#include <src/main/decaf/net/BindException.h>
```

Inheritance diagram for `decaf::net::BindException`:

Public Member Functions

- `BindException()` throw `()`

Default Constructor.

- **BindException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **BindException** (const **BindException** &ex) throw ()
Copy Constructor.
- **BindException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BindException** (const std::exception *cause) throw ()
Constructor.
- **BindException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **BindException** * **clone** () const
Clones this exception.
- virtual ~**BindException** () throw ()

6.135.1 Constructor & Destructor Documentation

6.135.1.1 decaf::net::BindException::BindException () throw () [inline]

Default Constructor.

6.135.1.2 decaf::net::BindException::BindException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.135.1.3 decaf::net::BindException::BindException (const BindException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.135.1.4 `decaf::net::BindException::BindException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.135.1.5 `decaf::net::BindException::BindException (const std::exception * cause) throw ()` [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.135.1.6 `decaf::net::BindException::BindException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.135.1.7 `virtual decaf::net::BindException::~~BindException () throw ()` [inline, virtual]

6.135.2 Member Function Documentation

6.135.2.1 `virtual BindException* decaf::net::BindException::clone () const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3300).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/BindException.h`

6.136 decaf::io::BlockingByteArrayInputStream Class Reference

This is a blocking version of a byte buffer stream.

```
#include <src/main/decaf/io/BlockingByteArrayInputStream.h>
```

Inheritance diagram for decaf::io::BlockingByteArrayInputStream:

Public Member Functions

- **BlockingByteArrayInputStream** ()
Default Constructor - uses a default internal buffer.
- **BlockingByteArrayInputStream** (const unsigned char *buffer, int bufferSize)
Constructor that initializes the internal buffer.
- virtual **~BlockingByteArrayInputStream** ()
- virtual void **setByteArray** (const unsigned char *buffer, int bufferSize)
- virtual int **available** () const throw (decaf::io::IOException)
Indicates the number of bytes available.
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.
The default implementation of this method returns zero.
Returns
the number of bytes available on this input stream.
Exceptions
IOException (p. 2003) if an I/O error occurs.
- virtual void **close** () throw (decaf::io::IOException)
*Closes the **InputStream** (p. 1909) freeing any resources that might have been acquired during the lifetime of this stream.*
The default implementation of this method does nothing.
- virtual long long **skip** (long long num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Skips over and discards n bytes of data from this input stream.

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

Parameters

***num** The number of bytes to skip.*

Returns

total bytes skipped

Exceptions

***IOException** (p. 2003) if an I/O error occurs.*

***UnsupportedOperationException** if the concrete stream class does not support skipping bytes.*

Protected Member Functions

- virtual int **doReadByte** () throw (**IOException**)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsExcpetion, decaf::lang::exceptions::NullPointerException)

6.136.1 Detailed Description

This is a blocking version of a byte buffer stream. Read operations block until the requested data becomes available in the internal buffer via a call to **setByteArray**.

6.136.2 Constructor & Destructor Documentation

6.136.2.1 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream ()

Default Constructor - uses a default internal buffer.

6.136.2.2 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream (const unsigned char * *buffer*, int *bufferSize*)

Constructor that initializes the internal buffer.

See also

setByteArray (p. 774).

6.136.2.3 `virtual`
`decaf::io::BlockingByteArrayInputStream::~~BlockingByteArrayInputStream`
`() [virtual]`

6.136.3 Member Function Documentation

6.136.3.1 `virtual int decaf::io::BlockingByteArrayInputStream::available () const`
`throw (decaf::io::IOException) [virtual]`

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented from `decaf::io::InputStream` (p. 1911).

6.136.3.2 `virtual void decaf::io::BlockingByteArrayInputStream::close () throw (`
`decaf::io::IOException) [virtual]`

Closes the **InputStream** (p. 1909) freeing any resources that might have been aquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from `decaf::io::InputStream` (p. 1912).

6.136.3.3 `virtual int de-`
`caf::io::BlockingByteArrayInputStream::doReadArrayBounded`
`(unsigned char * buffer, int size, int off-`
`set, int length) throw (decaf::io::IOException,`
`decaf::lang::exceptions::IndexOutOfBoundsException,`
`decaf::lang::exceptions::NullPointerException) [protected, virtual]`

Reimplemented from `decaf::io::InputStream` (p. 1912).

6.136.3.4 `virtual int decaf::io::BlockingByteArrayInputStream::doReadByte ()`
`throw (IOException) [protected, virtual]`

Implements `decaf::io::InputStream` (p. 1912).

6.136.3.5 virtual void decaf::io::BlockingByteArrayInputStream::setByteArray (const unsigned char * *buffer*, int *bufferSize*) [virtual]

6.136.3.6 virtual long long decaf::io::BlockingByteArrayInputStream::skip (long long *num*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1917).

The documentation for this class was generated from the following file:

- src/main/decaf/io/BlockingByteArrayInputStream.h

6.137 decaf::util::concurrent::BlockingQueue< E > Class Template Reference

A **decaf::util::Queue** (p. 2948) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

```
#include <src/main/decaf/util/concurrent/BlockingQueue.h>
```

Inheritance diagram for decaf::util::concurrent::BlockingQueue< E >:

Public Member Functions

- virtual ~BlockingQueue ()

- virtual void **put** (const E &value)=0 throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)

Inserts the specified element into this queue, waiting if necessary for space to become available.

- virtual bool **offer** (const E &e, long timeout, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)

Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

- virtual E **take** ()=0 throw (decaf::lang::exceptions::InterruptedException)

Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

- virtual bool **poll** (E &result, long long timeout, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::InterruptedException)

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

- virtual int **remainingCapacity** () const =0

*Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or **Integer::MAX_VALUE** if there is no intrinsic limit.*

- virtual std::size_t **drainTo** (**Collection**< E > &c)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)

Removes all available elements from this queue and adds them to the given collection.

- virtual std::size_t **drainTo** (**Collection**< E > &c, std::size_t maxElements)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)

Removes at most the given number of available elements from this queue and adds them to the given collection.

6.137.1 Detailed Description

`template<typename E> class decaf::util::concurrent::BlockingQueue< E >`

A **decaf::util::Queue** (p. 2948) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element. **BlockingQueue** (p. 774) methods come in four forms, with different ways of handling operations that cannot be satisfied immediately, but may be satisfied at some point in the future: one throws an exception, the second returns a special value (either **true** or **false**, depending on the operation), the third blocks the current thread indefinitely until the operation can succeed, and the fourth blocks for only a given maximum time limit before giving up. These methods are summarized in the following table:

	<i>Throws exception</i>	<i>Boolean Flag</i>	<i>Blocks</i>	<i>Times out</i>
Insert	add(e) (p. 159)	offer(e) (p. 779)	put(e) (p. 780)	offer(e, time, unit) (p. ??)
Remove	remove() (p. 161)	poll() (p. 779)	take() (p. 780)	poll(time, unit) (p. ??)
Examine	element() (p. 160)	peek() (p. 2950)	<i>not applicable</i>	<i>not applicable</i>

A `BlockingQueue` (p. 774) may be capacity bounded. At any given time it may have a `remainingCapacity` beyond which no additional elements can be `put` without blocking. A `BlockingQueue` (p. 774) without any intrinsic capacity constraints always reports a remaining capacity of `Integer::MAX_VALUE`.

`BlockingQueue` (p. 774) implementations are designed to be used primarily for producer-consumer queues, but additionally support `decaf::util::Collection` (p. 1097) interface. So, for example, it is possible to remove an arbitrary element from a queue using `remove(x)`. However, such operations are in general *not* performed very efficiently, and are intended for only occasional use, such as when a queued message is cancelled.

`BlockingQueue` (p. 774) implementations are thread-safe. All queuing methods achieve their effects atomically using internal locks or other forms of concurrency control. However, the *bulk* `Collection` (p. 1097) operations `addAll`, `containsAll`, `retainAll` and `removeAll` are *not* necessarily performed atomically unless specified otherwise in an implementation. So it is possible, for example, for `addAll(c)` to fail (throwing an exception) after adding only some of the elements in `c`.

A `BlockingQueue` (p. 774) does *not* intrinsically support any kind of "close" or "shutdown" operation to indicate that no more items will be added. The needs and usage of such features tend to be implementation-dependent. For example, a common tactic is for producers to insert special *end-of-stream* or *poison* objects, that are interpreted accordingly when taken by consumers.

Usage example, based on a typical producer-consumer scenario. Note that a `BlockingQueue` (p. 774) can safely be used with multiple producers and multiple consumers.

```
class Producer : public Runnable {
private:

    BlockingQueue* queue;

public:

    Producer( BlockingQueue* q ) : queue( q ) {}

    virtual void run() {
        try {
            while( true ) { queue->put( produce() ); }
        } catch( InterruptedException& ex ) { ... handle ...}
    }

    Object produce() { ... }
}

class Consumer : public Runnable {
private:

    BlockingQueue* queue;
```

```

public:

    Consumer( BlockingQueue* q ) : queue( q ) {}

    virtual void run() {
        try {
            while( true ) { consume( queue->take() (p.780) ); }
        } catch( InterruptedException& ex ) { ... handle ...}
    }

    void consume( Object& x ) { ... }
}

int main( int argc, char** argv ) {

    BlockingQueue (p.774) q = new SomeQueueImplementation();
    Producer p( &q );
    Consumer c1( &q );
    Consumer c2( &q );
    Thread t1( &p ).start();
    Thread t2( &c1 ).start();
    Thread t3( &c2 ).start();
}

```

Memory consistency effects: As with other concurrent collections, actions in a thread prior to placing an object into a **BlockingQueue** (p. 774) *happen-before* actions subsequent to the access or removal of that element from the **BlockingQueue** (p. 774) in another thread.

Since

1.0

6.137.2 Constructor & Destructor Documentation

6.137.2.1 `template<typename E > virtual decaf::util::concurrent::BlockingQueue< E >::~~BlockingQueue () [inline, virtual]`

6.137.3 Member Function Documentation

6.137.3.1 `template<typename E > virtual std::size_t decaf::util::concurrent::BlockingQueue< E >::drainTo (Collection< E > & c) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException) [pure virtual]`

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

c the collection to transfer elements into

Returns

the number of elements transferred

Exceptions

UnsupportedOperationException if addition of elements is not supported by the specified collection

IllegalArgumentException if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3481).

```
6.137.3.2  template<typename E > virtual std::size_t
            decaf::util::concurrent::BlockingQueue< E >::drainTo (
            Collection< E > & c, std::size_t maxElements ) throw
            ( decaf::lang::exceptions::UnsupportedOperationException,
            decaf::lang::exceptions::IllegalArgumentException ) [pure virtual]
```

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in **IllegalArgumentException**. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

c the collection to transfer elements into

maxElements the maximum number of elements to transfer

Returns

the number of elements transferred

Exceptions

UnsupportedOperationException if addition of elements is not supported by the specified collection

IllegalArgumentException if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3481).

6.137.3.3 `template<typename E > virtual bool
decaf::util::concurrent::BlockingQueue< E >::offer (
const E & e, long timeout, const TimeUnit & unit
) throw (decaf::lang::exceptions::InterruptedException,
decaf::lang::exceptions::NullPointerException, de-
caf::lang::exceptions::IllegalArgumentException) [pure
virtual]`

Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Parameters

e the element to add

timeout how long to wait before giving up, in units of *unit*

unit a TimeUnit (p. 3559) determining how to interpret the *timeout* parameter

Returns

`true` if successful, or `false` if the specified waiting time elapses before space is available

Exceptions

InterruptedException if interrupted while waiting

NullPointerException if the specified element is null

IllegalArgumentException if some property of the specified element prevents it from being added to this queue

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3483).

6.137.3.4 `template<typename E > virtual bool
decaf::util::concurrent::BlockingQueue< E >::poll (E &
result, long long timeout, const TimeUnit & unit) throw (
decaf::lang::exceptions::InterruptedException) [pure virtual]`

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

Parameters

result the referenced value that will be assigned the value retrieved from the **Queue** (p. 2948). Undefined if this methods returned false.

timeout how long to wait before giving up, in units of *unit*

unit a TimeUnit (p. 3559) determining how to interpret the *timeout* parameter.

Returns

`true` if successful or `false` if the specified waiting time elapses before an element is available.

Exceptions

InterruptedException if interrupted while waiting

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3484).

```

6.137.3.5  template<typename E > virtual void
           decaf::util::concurrent::BlockingQueue< E >::put ( const
           E & value ) throw ( decaf::lang::exceptions::InterruptedException,
           decaf::lang::exceptions::NullPointerException, de-
           ccaf::lang::exceptions::IllegalArgumentException ) [pure
           virtual]

```

Inserts the specified element into this queue, waiting if necessary for space to become available.

Parameters

value the element to add

Exceptions

InterruptedException if interrupted while waiting

NullPointerException if the specified element is null

IllegalArgumentException if some property of the specified element prevents it from being added to this queue

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3485).

```

6.137.3.6  template<typename E > virtual int
           decaf::util::concurrent::BlockingQueue< E >::remainingCapacity (    )
           const [pure virtual]

```

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is about to insert or remove an element.

Returns

the remaining capacity

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3485).

```

6.137.3.7  template<typename E > virtual E
           decaf::util::concurrent::BlockingQueue< E >::take (    )
           throw ( decaf::lang::exceptions::InterruptedException ) [pure virtual]

```

Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

Returns

the head of this queue

Exceptions

InterruptedException if interrupted while waiting

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3486).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/BlockingQueue.h`

6.138 decaf::lang::Boolean Class Reference

```
#include <src/main/decaf/lang/Boolean.h>
```

Inheritance diagram for `decaf::lang::Boolean`:

Public Member Functions

- **Boolean** (bool value)
- **Boolean** (const std::string &value)
- virtual **~Boolean** ()
- bool **booleanValue** () const
- std::string **toString** () const
- virtual int **compareTo** (const **Boolean** &b) const
*Compares this **Boolean** (p. 781) instance with another.*
- virtual bool **operator==** (const **Boolean** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Boolean** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const **Boolean** &b) const
- virtual int **compareTo** (const bool &b) const
*Compares this **Boolean** (p. 781) instance with another.*
- virtual bool **operator==** (const bool &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const bool &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const bool &b) const

Static Public Member Functions

- static **Boolean** **valueOf** (bool value)
- static **Boolean** **valueOf** (const std::string &value)
- static bool **parseBoolean** (const std::string &value)
*Parses the **String** (p. 3427) passed and extracts an bool.*

- static std::string **toString** (bool value)
*Converts the bool to a **String** (p. 3427) representation.*

Static Public Attributes

- static const **Boolean** **_FALSE**
The Class object representing the primitive false boolean.
- static const **Boolean** **_TRUE**
The Class object representing the primitive type boolean.

6.138.1 Constructor & Destructor Documentation

6.138.1.1 decaf::lang::Boolean::Boolean (bool value)

Parameters

value - primitive boolean to wrap.

6.138.1.2 decaf::lang::Boolean::Boolean (const std::string & value)

Parameters

value - **String** (p. 3427) value to convert to a boolean.

6.138.1.3 virtual decaf::lang::Boolean::~~Boolean () [inline, virtual]

6.138.2 Member Function Documentation

6.138.2.1 bool decaf::lang::Boolean::booleanValue () const [inline]

Returns

the primitive boolean value of this object

6.138.2.2 virtual int decaf::lang::Boolean::compareTo (const Boolean & b) const [virtual]

Compares this **Boolean** (p. 781) instance with another.

Parameters

b - the **Boolean** (p. 781) instance to be compared

Returns

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

6.138.2.3 `virtual int decaf::lang::Boolean::compareTo (const bool & b) const`
[virtual]

Compares this **Boolean** (p. 781) instance with another.

Parameters

b - the **Boolean** (p. 781) instance to be compared

Returns

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements **decaf::lang::Comparable**< **bool** > (p. 1126).

6.138.2.4 `bool decaf::lang::Boolean::equals (const bool & b) const` [inline,
virtual]

Returns

true if the two **Boolean** (p. 781) Objects have the same value.

Implements **decaf::lang::Comparable**< **bool** > (p. 1126).

6.138.2.5 `bool decaf::lang::Boolean::equals (const Boolean & b) const` [inline]

Returns

true if the two **Boolean** (p. 781) Objects have the same value.

6.138.2.6 `virtual bool decaf::lang::Boolean::operator< (const bool & value)`
`const` [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **bool** > (p. 1127).

6.138.2.7 `virtual bool decaf::lang::Boolean::operator< (const Boolean & value)`
`const` [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.138.2.8 `virtual bool decaf::lang::Boolean::operator==(const bool & value) const [virtual]`

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< bool >` (p. 1127).

6.138.2.9 `virtual bool decaf::lang::Boolean::operator==(const Boolean & value) const [virtual]`

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.138.2.10 `static bool decaf::lang::Boolean::parseBoolean (const std::string & value) [static]`

Parses the `String` (p. 3427) passed and extracts an bool.

Parameters

value The std::string value to parse

Returns

bool value

6.138.2.11 `static std::string decaf::lang::Boolean::toString (bool value) [static]`

Converts the bool to a **String** (p. 3427) representation.

Parameters

value The bool value to convert.

Returns

std::string representation of the bool value passed.

6.138.2.12 `std::string decaf::lang::Boolean::toString () const`**Returns**

the string representation of this Booleans value.

6.138.2.13 `static Boolean decaf::lang::Boolean::valueOf (bool value) [static]`**Parameters**

value The bool value to convert to a Boolean (p. 781) instance.

Returns

a **Boolean** (p. 781) instance of the primitive boolean value

6.138.2.14 `static Boolean decaf::lang::Boolean::valueOf (const std::string & value) [static]`**Parameters**

value The std::string value to convert to a Boolean (p. 781) instance.

Returns

a **Boolean** (p. 781) instance of the string value

6.138.3 Field Documentation**6.138.3.1** `const Boolean decaf::lang::Boolean::_FALSE [static]`

The Class object representing the primitive false boolean.

6.138.3.2 `const Boolean decaf::lang::Boolean::_TRUE [static]`

The Class object representing the primitive type boolean.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Boolean.h**

6.139 activemq::commands::BooleanExpression Class Reference

```
#include <src/main/activemq/commands/BooleanExpression.h>
```

Inheritance diagram for activemq::commands::BooleanExpression:

Public Member Functions

- **BooleanExpression** ()
- virtual **~BooleanExpression** ()
- virtual **DataStructure * cloneDataStructure** () const
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src AMQCPP_UNUSED)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

6.139.1 Constructor & Destructor Documentation

- 6.139.1.1** **activemq::commands::BooleanExpression::BooleanExpression** ()
[inline]
- 6.139.1.2** **virtual activemq::commands::BooleanExpression::~~BooleanExpression** ()
[inline, virtual]

6.139.2 Member Function Documentation

- 6.139.2.1** **virtual DataStructure* activemq::commands::BooleanExpression::cloneDataStructure** () const
[inline, virtual]

Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.139.2.2 `virtual void activemq::commands::BooleanExpression::copyDataStructure (const DataStructure *src AMQCPP_UNUSED) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::BaseDataStructure` (p. 766).

6.139.2.3 `virtual bool activemq::commands::BooleanExpression::equals (const DataStructure * value) const [inline, virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1556).

References `activemq::commands::BaseDataStructure::equals()`.

6.139.2.4 `virtual std::string activemq::commands::BooleanExpression::toString () const [inline, virtual]`

Returns a string containing the information for this `DataStructure` (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 767).

References `activemq::commands::BaseDataStructure::toString()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BooleanExpression.h`

6.140 `activemq::wireformat::openwire::utils::BooleanStream` Class Reference

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

```
#include <src/main/activemq/wireformat/openwire/utils/BooleanStream.h>
```

Public Member Functions

- **BooleanStream** ()
- virtual **~BooleanStream** ()
- bool **readBoolean** () throw (decaf::io::IOException)
Read a boolean data element from the internal data buffer.
- void **writeBoolean** (bool value) throw (decaf::io::IOException)
Writes a Boolean value to the internal data buffer.
- void **marshal** (decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Marshal the data to a DataOutputStream.
- void **marshal** (std::vector< unsigned char > &dataOut)
Marshal the data to a STL vector of unsigned chars.
- void **unmarshal** (decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Unmarshal a Boolean data stream from the Input Stream.
- void **clear** ()
Clears to old position markers, data starts at the beginning.
- int **marshalledSize** ()
Calc the size that data is marshalled to.

6.140.1 Detailed Description

Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream. The booleans are stored as single bits in the stream, with the stream size pre-pended to the stream when the data is marshalled.

The serialized form of the size field can be between 1 and 3 bytes. If the number of used bytes < 64, size=1 byte If the number of used bytes >=64 and < 256 (size of an unsigned byte), size=2 bytes If the number of used bytes >=256, size=3 bytes

The high-order 2 bits (128 and 64) of the first byte of the size field are used to encode the information about the number of bytes in the size field. The only time the first byte will contain a value >=64 is if there are more bytes in the size field. If the first byte < 64, the value of the byte is simply the size value. If the first byte = 0xC0, the following unsigned byte is the size field. If the first byte = 0x80, the following short (two bytes) are the size field.

6.140.2 Constructor & Destructor Documentation

6.140.2.1 `activemq::wireformat::openwire::utils::BooleanStream::BooleanStream ()`

6.140.2.2 `virtual
activemq::wireformat::openwire::utils::BooleanStream::~~BooleanStream ()` [inline, virtual]

6.140.3 Member Function Documentation

6.140.3.1 `void activemq::wireformat::openwire::utils::BooleanStream::clear ()`

Clears to old position markers, data starts at the beginning.

6.140.3.2 `void activemq::wireformat::openwire::utils::BooleanStream::marshal (std::vector< unsigned char > & dataOut)`

Marshal the data to a STL vector of unsigned chars.

Parameters

dataOut - reference to a vector to write the data to.

6.140.3.3 `void activemq::wireformat::openwire::utils::BooleanStream::marshal (decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)`

Marshal the data to a DataOutputStream.

Parameters

dataOut - Stream to write the data to.

6.140.3.4 `int activemq::wireformat::openwire::utils::BooleanStream::marshalledSize ()`

Calc the size that data is marshalled to.

Returns

int size of marshalled data.

6.140.3.5 `bool activemq::wireformat::openwire::utils::BooleanStream::readBoolean () throw (decaf::io::IOException)`

Read a boolean data element from the internal data buffer.

Returns

boolean from the stream

6.140.3.6 `void activemq::wireformat::openwire::utils::BooleanStream::unmarshal (decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)`

Unmarshal a Boolean data stream from the Input Stream.

Parameters

dataIn - Input Stream to read data from.

6.140.3.7 `void activemq::wireformat::openwire::utils::BooleanStream::writeBoolean (bool value) throw (decaf::io::IOException)`

Writes a Boolean value to the internal data buffer.

Parameters

value - boolean data to write.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/BooleanStream.h`

6.141 decaf::util::concurrent::BrokenBarrierException Class Reference

```
#include <src/main/decaf/util/concurrent/BrokenBarrierException.h>
```

Inheritance diagram for decaf::util::concurrent::BrokenBarrierException:

Public Member Functions

- **BrokenBarrierException** () throw ()
Default Constructor.
- **BrokenBarrierException** (const decaf::lang::Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **BrokenBarrierException** (const BrokenBarrierException &ex) throw ()
Copy Constructor.
- **BrokenBarrierException** (const std::exception *cause) throw ()
Constructor.
- **BrokenBarrierException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.

- **BrokenBarrierException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **BrokenBarrierException** * clone () const

Clones this exception.

- virtual ~**BrokenBarrierException** () throw ()

6.141.1 Constructor & Destructor Documentation

6.141.1.1 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException () throw () [inline]

Default Constructor.

6.141.1.2 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.141.1.3 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const BrokenBarrierException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

6.141.1.4 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const std::exception * cause) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.141.1.5 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

msg - The message to report

... - list of primitives that are formatted into the message

6.141.1.6 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

cause - The exception that was the cause for this one to be thrown.

msg - The message to report

... - list of primitives that are formatted into the message

6.141.1.7 virtual decaf::util::concurrent::BrokenBarrierException::~~BrokenBarrierException () throw () [inline, virtual]

6.141.2 Member Function Documentation

6.141.2.1 virtual BrokenBarrierException* decaf::util::concurrent::BrokenBarrierException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**BrokenBarrierException.h**

6.142 activemq::commands::BrokerError Class Reference

This class represents an Exception sent from the Broker.

```
#include <src/main/activemq/commands/BrokerError.h>
```

Inheritance diagram for activemq::commands::BrokerError:

Data Structures

- struct **StackTraceElement**

Public Member Functions

- **BrokerError** ()
- virtual **~BrokerError** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1553) Type as defined in CommandTypes.h.*
- virtual **BrokerError** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- virtual const std::string & **getMessage** () const
Gets the string holding the error message.
- virtual void **setMessage** (const std::string &message)
*Sets the string that contains the error **Message** (p. 2358).*
- virtual const std::string & **getExceptionClass** () const
Gets the string holding the Exception Class name.
- virtual void **setExceptionClass** (const std::string &exceptionClass)
Sets the string that contains the Exception Class name.
- virtual const **decaf::lang::Pointer**< **BrokerError** > & **getCause** () const
Gets the Broker Error that caused this exception.
- virtual void **setCause** (const **decaf::lang::Pointer**< **BrokerError** > &cause)
Sets the Broker Error that caused this exception.

- virtual const std::vector< **decaf::lang::Pointer< StackTraceElement >** > & **getStackTraceElements** () const

Gets the Stack Trace Elements for the Exception.

- virtual void **setStackTraceElements** (const std::vector< **decaf::lang::Pointer< StackTraceElement >** > &stackTraceElements)

Sets the Stack Trace Elements for this Exception.

6.142.1 Detailed Description

This class represents an Exception sent from the Broker. The Broker sends java Throwables, so we must mimic its structure here.

6.142.2 Constructor & Destructor Documentation

6.142.2.1 **activemq::commands::BrokerError::BrokerError** ()

6.142.2.2 **virtual activemq::commands::BrokerError::~~BrokerError** () [virtual]

6.142.3 Member Function Documentation

6.142.3.1 **virtual BrokerError* activemq::commands::BrokerError::cloneDataStructure** () const [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

References **copyDataStructure()**.

6.142.3.2 **virtual void activemq::commands::BrokerError::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

Referenced by **cloneDataStructure()**.

6.142.3.3 `virtual const decaf::lang::Pointer<BrokerError>& activemq::commands::BrokerError::getCause () const [inline, virtual]`

Gets the Broker Error that caused this exception.

Returns

Broker Error Pointer

6.142.3.4 `virtual unsigned char activemq::commands::BrokerError::getDataStructureType () const [inline, virtual]`

Get the **DataStructure** (p.1553) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p.1557).

6.142.3.5 `virtual const std::string& activemq::commands::BrokerError::getExceptionClass () const [inline, virtual]`

Gets the string holding the Exception Class name.

Returns

Exception Class name

6.142.3.6 `virtual const std::string& activemq::commands::BrokerError::getMessage () const [inline, virtual]`

Gets the string holding the error message.

Returns

String **Message** (p.2358)

6.142.3.7 `virtual const std::vector< decaf::lang::Pointer<StackTraceElement> >& activemq::commands::BrokerError::getStackTraceElements () const [inline, virtual]`

Gets the Stack Trace Elements for the Exception.

Returns

Stack Trace Elements

6.142.3.8 `virtual void activemq::commands::BrokerError::setCause (const decaf::lang::Pointer< BrokerError > & cause) [inline, virtual]`

Sets the Broker Error that caused this exception.

Parameters

cause - Broker Error

6.142.3.9 `virtual void activemq::commands::BrokerError::setExceptionClass (const std::string & exceptionClass) [inline, virtual]`

Sets the string that contains the Exception Class name.

Parameters

exceptionClass - String Exception Class name

6.142.3.10 `virtual void activemq::commands::BrokerError::setMessage (const std::string & message) [inline, virtual]`

Sets the string that contains the error **Message** (p.2358).

Parameters

message - String Error **Message** (p.2358)

6.142.3.11 `virtual void activemq::commands::BrokerError::setStackTraceElements (const std::vector< decaf::lang::Pointer< StackTraceElement > > & stackTraceElements) [inline, virtual]`

Sets the Stack Trace Elements for this Exception.

Parameters

stackTraceElements - Stack Trace Elements

6.142.3.12 `virtual decaf::lang::Pointer<commands::Command> activemq::commands::BrokerError::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p.3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1112).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerError.h`

6.143 **activemq::exceptions::BrokerException** Class Reference

```
#include <src/main/activemq/exceptions/BrokerException.h>
```

Inheritance diagram for **activemq::exceptions::BrokerException**:

Public Member Functions

- **BrokerException** () throw ()
- **BrokerException** (const **exceptions::ActiveMQException** &ex) throw ()
- **BrokerException** (const **BrokerException** &ex) throw ()
- **BrokerException** (const char *file, const int lineNumber, const char *msg,...) throw ()
- **BrokerException** (const char *file, const int lineNumber, const **commands::BrokerError** *error) throw ()
- virtual **BrokerException** * **clone** () const
Clones this exception.
- virtual ~**BrokerException** () throw ()

6.143.1 Constructor & Destructor Documentation

6.143.1.1 **activemq::exceptions::BrokerException::BrokerException** () throw ()
[inline]

6.143.1.2 **activemq::exceptions::BrokerException::BrokerException** (const **exceptions::ActiveMQException** & *ex*) throw () [inline]

6.143.1.3 **activemq::exceptions::BrokerException::BrokerException** (const **BrokerException** & *ex*) throw () [inline]

6.143.1.4 **activemq::exceptions::BrokerException::BrokerException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
[inline]

6.143.1.5 **activemq::exceptions::BrokerException::BrokerException** (const char * *file*, const int *lineNumber*, const **commands::BrokerError** * *error*) throw () [inline]

References `decaf::lang::Exception::setMark()`, and `decaf::lang::Exception::setMessage()`.

6.143.1.6 virtual `activemq::exceptions::BrokerException::~~BrokerException ()`
`throw ()` [inline, virtual]

6.143.2 Member Function Documentation

6.143.2.1 virtual `BrokerException* activemq::exceptions::BrokerException::clone ()`
`const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from `activemq::exceptions::ActiveMQException` (p. 315).

The documentation for this class was generated from the following file:

- `src/main/activemq/exceptions/BrokerException.h`

6.144 activemq::commands::BrokerId Class Reference

```
#include <src/main/activemq/commands/BrokerId.h>
```

Inheritance diagram for `activemq::commands::BrokerId`:

Public Types

- typedef `decaf::lang::PointerComparator< BrokerId > COMPARATOR`

Public Member Functions

- `BrokerId ()`
- `BrokerId (const BrokerId &other)`
- virtual `~BrokerId ()`
- virtual unsigned char `getDataStructureType ()` const
Get the unique identifier that this object and its own Marshaler share.
- virtual `BrokerId * cloneDataStructure ()` const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void `copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string `toString ()` const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool `equals (const DataStructure *value)` const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &**value**)
- virtual int **compareTo** (const **BrokerId** &**value**) const
- virtual bool **equals** (const **BrokerId** &**value**) const
- virtual bool **operator==** (const **BrokerId** &**value**) const
- virtual bool **operator<** (const **BrokerId** &**value**) const
- **BrokerId** & **operator=** (const **BrokerId** &other)

Static Public Attributes

- static const unsigned char **ID_BROKERID** = 124

Protected Attributes

- std::string **value**

6.144.1 Member Typedef Documentation

- 6.144.1.1** `typedef decaf::lang::PointerComparator<BrokerId>`
 `activemq::commands::BrokerId::COMPARATOR`

6.144.2 Constructor & Destructor Documentation

- 6.144.2.1** `activemq::commands::BrokerId::BrokerId ()`
- 6.144.2.2** `activemq::commands::BrokerId::BrokerId (const BrokerId & other)`
- 6.144.2.3** `virtual activemq::commands::BrokerId::~~BrokerId () [virtual]`

6.144.3 Member Function Documentation

- 6.144.3.1** `virtual BrokerId* activemq::commands::BrokerId::cloneDataStructure (`
 `) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

- 6.144.3.2** `virtual int activemq::commands::BrokerId::compareTo (const BrokerId`
 `& value) const [virtual]`
- 6.144.3.3** `virtual void activemq::commands::BrokerId::copyDataStructure (const`
 `DataSeture * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.144.3.4 `virtual bool activemq::commands::BrokerId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.144.3.5 `virtual bool activemq::commands::BrokerId::equals (const BrokerId & value) const` [virtual]

6.144.3.6 `virtual unsigned char activemq::commands::BrokerId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

6.144.3.7 `virtual const std::string& activemq::commands::BrokerId::getValue () const` [virtual]

6.144.3.8 `virtual std::string& activemq::commands::BrokerId::getValue ()` [virtual]

6.144.3.9 `virtual bool activemq::commands::BrokerId::operator< (const BrokerId & value) const` [virtual]

6.144.3.10 `BrokerId& activemq::commands::BrokerId::operator= (const BrokerId & other)`

6.144.3.11 `virtual bool activemq::commands::BrokerId::operator== (const BrokerId & value) const` [virtual]

6.144.3.12 `virtual void activemq::commands::BrokerId::setValue (const std::string & value)` [virtual]

6.144.3.13 `virtual std::string activemq::commands::BrokerId::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.144.4 Field Documentation

6.144.4.1 `const unsigned char activemq::commands::BrokerId::ID_BROKERID = 124 [static]`

6.144.4.2 `std::string activemq::commands::BrokerId::value [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerId.h`

6.145 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 801).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller`:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.145.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 801). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.145.2 Constructor & Destructor Documentation

6.145.2.1 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::BrokerIdMarshaller () [inline]

6.145.2.2 virtual activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::~~BrokerIdMarshaller () [inline, virtual]

6.145.3 Member Function Documentation

6.145.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.145.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.145.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.145.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.145.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.145.3.6 virtual void **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal2**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.145.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h`

6.146 **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 805).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller**:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.146.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 805). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.146.2 Constructor & Destructor Documentation

6.146.2.1 `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::BrokerIdMarshaller () [inline]`

6.146.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::~~BrokerIdMarshaller () [inline, virtual]`

6.146.3 Member Function Documentation

6.146.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.146.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::getDataStructureType
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.146.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

```
6.146.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

```
6.146.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.146.3.6 virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.146.3.7 virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**BrokerIdMarshaller.h**

6.147 activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 808).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller`:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.147.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 808). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.147.2 Constructor & Destructor Documentation

6.147.2.1 `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::BrokerIdMarshaller () [inline]`

6.147.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::~~BrokerIdMarshaller () [inline, virtual]`

6.147.3 Member Function Documentation

6.147.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.147.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.147.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.147.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.147.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.147.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

```
6.147.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h`

6.148 `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerIdMarshaller` (p. 812).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller`:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.148.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 812). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.148.2 Constructor & Destructor Documentation

6.148.2.1 `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::BrokerIdMarshaller () [inline]`

6.148.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::~~BrokerIdMarshaller
() [inline, virtual]`

6.148.3 Member Function Documentation

6.148.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.148.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::getDataStructureType
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.148.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.148.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.148.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.148.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

```
6.148.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h`

6.149 `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerIdMarshaller` (p. 816).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller`:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.149.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 816). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.149.2 Constructor & Destructor Documentation

6.149.2.1 `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::BrokerIdMarshaller () [inline]`

6.149.2.2 `virtual
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::~~BrokerIdMarshaller
() [inline, virtual]`

6.149.3 Member Function Documentation

6.149.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.149.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::getDataStructureType
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.149.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.149.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.149.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.149.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

```
6.149.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h`

6.150 `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerIdMarshaller` (p. 820).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller`:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.150.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 820). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.150.2 Constructor & Destructor Documentation

6.150.2.1 `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::BrokerIdMarshaller () [inline]`

6.150.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::~~BrokerIdMarshaller
() [inline, virtual]`

6.150.3 Member Function Documentation

6.150.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.150.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::getDataStructureType
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.150.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.150.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.150.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.150.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.150.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**BrokerIdMarshaller.h**

6.151 activemq::commands::BrokerInfo Class Reference

```
#include <src/main/activemq/commands/BrokerInfo.h>
```

Inheritance diagram for **activemq::commands::BrokerInfo**:

Public Member Functions

- **BrokerInfo** ()
- virtual **~BrokerInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **BrokerInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerId** > & **getBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getBrokerId** ()
- virtual void **setBrokerId** (const **Pointer**< **BrokerId** > &brokerId)
- virtual const std::string & **getBrokerURL** () const
- virtual std::string & **getBrokerURL** ()
- virtual void **setBrokerURL** (const std::string &brokerURL)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** ()
- virtual void **setPeerBrokerInfos** (const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > &peerBrokerInfos)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)
- virtual bool **isSlaveBroker** () const
- virtual void **setSlaveBroker** (bool slaveBroker)
- virtual bool **isMasterBroker** () const
- virtual void **setMasterBroker** (bool masterBroker)
- virtual bool **isFaultTolerantConfiguration** () const
- virtual void **setFaultTolerantConfiguration** (bool faultTolerantConfiguration)
- virtual bool **isDuplexConnection** () const
- virtual void **setDuplexConnection** (bool duplexConnection)
- virtual bool **isNetworkConnection** () const
- virtual void **setNetworkConnection** (bool networkConnection)
- virtual long long **getConnectionId** () const
- virtual void **setConnectionId** (long long connectionId)
- virtual const std::string & **getBrokerUploadUrl** () const
- virtual std::string & **getBrokerUploadUrl** ()
- virtual void **setBrokerUploadUrl** (const std::string &brokerUploadUrl)
- virtual const std::string & **getNetworkProperties** () const
- virtual std::string & **getNetworkProperties** ()
- virtual void **setNetworkProperties** (const std::string &networkProperties)
- virtual bool **isBrokerInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor)
throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_BROKERINFO** = 2

Protected Attributes

- **Pointer< BrokerId > brokerId**
- **std::string brokerURL**
- **std::vector< decaf::lang::Pointer< BrokerInfo > > peerBrokerInfos**
- **std::string brokerName**
- **bool slaveBroker**
- **bool masterBroker**
- **bool faultTolerantConfiguration**
- **bool duplexConnection**
- **bool networkConnection**
- **long long connectionId**
- **std::string brokerUploadUrl**
- **std::string networkProperties**

6.151.1 Constructor & Destructor Documentation

6.151.1.1 **activemq::commands::BrokerInfo::BrokerInfo ()**

6.151.1.2 **virtual activemq::commands::BrokerInfo::~~BrokerInfo () [virtual]**

6.151.2 Member Function Documentation

6.151.2.1 **virtual BrokerInfo* activemq::commands::BrokerInfo::cloneDataStructure () const [virtual]**

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.151.2.2 **virtual void activemq::commands::BrokerInfo::copyDataStructure (const DataStructure * *src*) [virtual]**

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.151.2.3 `virtual bool activemq::commands::BrokerInfo::equals (const
DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.151.2.4 `virtual const Pointer<BrokerId>& ac-
tivemq::commands::BrokerInfo::getBrokerId () const`
[virtual]

6.151.2.5 `virtual Pointer<BrokerId>& ac-
tivemq::commands::BrokerInfo::getBrokerId ()`
[virtual]

6.151.2.6 `virtual std::string& activemq::commands::BrokerInfo::getBrokerName ()`
[virtual]

6.151.2.7 `virtual const std::string& ac-
tivemq::commands::BrokerInfo::getBrokerName ()`
const [virtual]

6.151.2.8 `virtual const std::string& ac-
tivemq::commands::BrokerInfo::getBrokerUploadUrl ()`
const [virtual]

6.151.2.9 `virtual std::string& ac-
tivemq::commands::BrokerInfo::getBrokerUploadUrl ()`
[virtual]

6.151.2.10 `virtual const std::string& ac-
tivemq::commands::BrokerInfo::getBrokerURL ()`
const [virtual]

6.151.2.11 `virtual std::string& activemq::commands::BrokerInfo::getBrokerURL ()`
[virtual]

6.151.2.12 `virtual long long activemq::commands::BrokerInfo::getConnectionId ()`
const [virtual]

6.151.2.13 `virtual unsigned char ac-
tivemq::commands::BrokerInfo::getDataStructureType ()`
const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1553) type copy.

Implements **activemq::commands::DataSet** (p. 1557).

6.151.2.14 virtual const std::string& activemq::commands::BrokerInfo::getNetworkProperties () const [virtual]

6.151.2.15 virtual std::string& activemq::commands::BrokerInfo::getNetworkProperties () [virtual]

6.151.2.16 virtual const std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos () const [virtual]

6.151.2.17 virtual std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos () [virtual]

6.151.2.18 virtual bool activemq::commands::BrokerInfo::isBrokerInfo () const [inline, virtual]

Returns

an answer of true to the **isBrokerInfo()** (p. 828) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 698).

- 6.151.2.19 `virtual bool activemq::commands::BrokerInfo::isDuplexConnection () const [virtual]`
- 6.151.2.20 `virtual bool activemq::commands::BrokerInfo::isFaultTolerantConfiguration () const [virtual]`
- 6.151.2.21 `virtual bool activemq::commands::BrokerInfo::isMasterBroker () const [virtual]`
- 6.151.2.22 `virtual bool activemq::commands::BrokerInfo::isNetworkConnection () const [virtual]`
- 6.151.2.23 `virtual bool activemq::commands::BrokerInfo::isSlaveBroker () const [virtual]`
- 6.151.2.24 `virtual void activemq::commands::BrokerInfo::setBrokerId (const Pointer< BrokerId > & brokerId) [virtual]`
- 6.151.2.25 `virtual void activemq::commands::BrokerInfo::setBrokerName (const std::string & brokerName) [virtual]`
- 6.151.2.26 `virtual void activemq::commands::BrokerInfo::setBrokerUploadUrl (const std::string & brokerUploadUrl) [virtual]`
- 6.151.2.27 `virtual void activemq::commands::BrokerInfo::setBrokerURL (const std::string & brokerURL) [virtual]`
- 6.151.2.28 `virtual void activemq::commands::BrokerInfo::setConnectionId (long long connectionId) [virtual]`
- 6.151.2.29 `virtual void activemq::commands::BrokerInfo::setDuplexConnection (bool duplexConnection) [virtual]`
- 6.151.2.30 `virtual void activemq::commands::BrokerInfo::setFaultTolerantConfiguration (bool faultTolerantConfiguration) [virtual]`
- 6.151.2.31 `virtual void activemq::commands::BrokerInfo::setMasterBroker (bool masterBroker) [virtual]`
- 6.151.2.32 `virtual void activemq::commands::BrokerInfo::setNetworkConnection (bool networkConnection) [virtual]`
- 6.151.2.33 `virtual void activemq::commands::BrokerInfo::setNetworkProperties (const std::string & networkProperties) [virtual]`
- 6.151.2.34 `virtual void activemq::commands::BrokerInfo::setPeerBrokerInfos (const std::vector< decaf::lang::Pointer< BrokerInfo > > & peerBrokerInfos) [virtual]`
- 6.151.2.35 `virtual void activemq::commands::BrokerInfo::setSlaveBroker (bool slaveBroker) [virtual]`
- 6.151.2.36 `virtual std::string activemq::commands::BrokerInfo::toString () const [virtual]`

Generated on Sun Sep 11 2011 18:23:35 for activemq-cpp-3.2.1 by Doxygen

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

6.151.2.37 `virtual Pointer<Command> activemq::commands::BrokerInfo::visit
(activemq::state::CommandVisitor * visitor) throw (
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.151.3 Field Documentation

- 6.151.3.1 `Pointer<BrokerId> activemq::commands::BrokerInfo::brokerId` [protected]
- 6.151.3.2 `std::string activemq::commands::BrokerInfo::brokerName` [protected]
- 6.151.3.3 `std::string activemq::commands::BrokerInfo::brokerUploadUrl` [protected]
- 6.151.3.4 `std::string activemq::commands::BrokerInfo::brokerURL` [protected]
- 6.151.3.5 `long long activemq::commands::BrokerInfo::connectionId` [protected]
- 6.151.3.6 `bool activemq::commands::BrokerInfo::duplexConnection` [protected]
- 6.151.3.7 `bool activemq::commands::BrokerInfo::faultTolerantConfiguration` [protected]
- 6.151.3.8 `const unsigned char activemq::commands::BrokerInfo::ID _ - BROKERINFO = 2` [static]
- 6.151.3.9 `bool activemq::commands::BrokerInfo::masterBroker` [protected]
- 6.151.3.10 `bool activemq::commands::BrokerInfo::networkConnection` [protected]
- 6.151.3.11 `std::string activemq::commands::BrokerInfo::networkProperties` [protected]
- 6.151.3.12 `std::vector< decaf::lang::Pointer<BrokerInfo> > activemq::commands::BrokerInfo::peerBrokerInfos` [protected]
- 6.151.3.13 `bool activemq::commands::BrokerInfo::slaveBroker` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerInfo.h`

6.152 `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerInfoMarshaller` (p. 831).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller`:

Public Member Functions

- `BrokerInfoMarshaller ()`

- virtual `~BrokerInfoMarshaller ()`
- virtual `commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- virtual unsigned char `getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.152.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 831). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.152.2 Constructor & Destructor Documentation

6.152.2.1 `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::BrokerInfoMarshaller () [inline]`

6.152.2.2 `virtual activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::~~BrokerInfoMarshaller () [inline, virtual]`

6.152.3 Member Function Documentation

6.152.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.152.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.152.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.152.3.4 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 703).

6.152.3.5 virtual int activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 704).

6.152.3.6 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 706).

6.152.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h`

6.153 **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 835).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller**:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.153.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 835). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.153.2 Constructor & Destructor Documentation

6.153.2.1 `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

6.153.2.2 `virtual activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

6.153.3 Member Function Documentation

6.153.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.153.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.153.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.153.3.4 virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 710).

6.153.3.5 virtual int activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 711).

6.153.3.6 virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 712).

6.153.3.7 virtual void **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h`

6.154 **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 839).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller**:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.154.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 839). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.154.2 Constructor & Destructor Documentation

6.154.2.1 `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::BrokerInfoMarshaller () [inline]`

6.154.2.2 `virtual activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::~~BrokerInfoMarshaller () [inline, virtual]`

6.154.3 Member Function Documentation

6.154.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.154.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.154.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.154.3.4 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.154.3.5 virtual int activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

6.154.3.6 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

6.154.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h`

6.155 `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerInfoMarshaller` (p. 843).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller`:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.155.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 843). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.155.2 Constructor & Destructor Documentation

6.155.2.1 `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::BrokerInfoMarshaller () [inline]`

6.155.2.2 `virtual activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::~~BrokerInfoMarshaller () [inline, virtual]`

6.155.3 Member Function Documentation

6.155.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.155.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.155.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.155.3.4 virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 723).

6.155.3.5 virtual int activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 724).

6.155.3.6 virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

6.155.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h`

6.156 `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerInfoMarshaller` (p. 847).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller`:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.156.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 847). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.156.2 Constructor & Destructor Documentation

6.156.2.1 `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

6.156.2.2 `virtual activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

6.156.3 Member Function Documentation

6.156.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.156.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.156.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.156.3.4 virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 730).

6.156.3.5 virtual int activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 731).

6.156.3.6 virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

```
6.156.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h`

6.157 `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerInfoMarshaller` (p. 851).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller`:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.157.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 851). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.157.2 Constructor & Destructor Documentation

6.157.2.1 `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::BrokerInfoMarshaller () [inline]`

6.157.2.2 `virtual activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::~~BrokerInfoMarshaller () [inline, virtual]`

6.157.3 Member Function Documentation

6.157.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.157.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.157.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.157.3.4 virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 737).

6.157.3.5 virtual int activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 738).

6.157.3.6 virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

```
6.157.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h`

6.158 decaf::nio::Buffer Class Reference

A container for data of a specific primitive type.

```
#include <src/main/decaf/nio/Buffer.h>
```

Inheritance diagram for `decaf::nio::Buffer`:

Public Member Functions

- **Buffer** (int capacity)
- **Buffer** (const **Buffer** &other)
- virtual ~**Buffer** ()
- virtual int **capacity** () const
- virtual int **position** () const
- virtual **Buffer** & **position** (int newPosition) throw (lang::exceptions::IllegalArgumentException)
Sets this buffer's position.
- virtual int **limit** () const
- virtual **Buffer** & **limit** (int newLimit) throw (lang::exceptions::IllegalArgumentException)
Sets this buffer's limit.
- virtual **Buffer** & **mark** ()
Sets this buffer's mark at its position.
- virtual **Buffer** & **reset** () throw (InvalidMarkException)
Resets this buffer's position to the previously-marked position.
- virtual **Buffer** & **clear** ()
Clears this buffer.
- virtual **Buffer** & **flip** ()
Flips this buffer.
- virtual **Buffer** & **rewind** ()
Rewinds this buffer.
- virtual int **remaining** () const
Returns the number of elements between the current position and the limit.
- virtual bool **hasRemaining** () const
Tells whether there are any elements between the current position and the limit.
- virtual bool **isReadOnly** () const =0
Tells whether or not this buffer is read-only.

Protected Attributes

- int **_position**
- int **_capacity**
- int **_limit**
- int **_mark**
- bool **_markSet**

6.158.1 Detailed Description

A container for data of a specific primitive type. A buffer is a linear, finite sequence of elements of a specific primitive type. Aside from its content, the essential properties of a buffer are its capacity, limit, and position:

A buffer's capacity is the number of elements it contains. The capacity of a buffer is never negative and never changes.

A buffer's limit is the index of the first element that should not be read or written. A buffer's limit is never negative and is never greater than its capacity.

A buffer's position is the index of the next element to be read or written. A buffer's position is never negative and is never greater than its limit.

There is one subclass of this class for each non-boolean primitive type.

Transferring data: Each subclass of this class defines two categories of get and put operations: * Relative operations read or write one or more elements starting at the current position and then increment the position by the number of elements transferred. If the requested transfer exceeds the limit then a relative get operation throws a **BufferUnderflowException** (p. 882) and a relative put operation throws a **BufferOverflowException** (p. 880); in either case, no data is transferred. * Absolute operations take an explicit element index and do not affect the position. Absolute get and put operations throw an **IndexOutOfBoundsException** if the index argument exceeds the limit.

Data may also, of course, be transferred in to or out of a buffer by the I/O operations of an appropriate channel, which are always relative to the current position.

Marking and resetting:

A buffer's mark is the index to which its position will be reset when the reset method is invoked. The mark is not always defined, but when it is defined it is never negative and is never greater than the position. If the mark is defined then it is discarded when the position or the limit is adjusted to a value smaller than the mark. If the mark is not defined then invoking the reset method causes an **InvalidMarkException** (p. 1997) to be thrown.

Invariants:

The following invariant holds for the mark, position, limit, and capacity values: $0 \leq \text{mark} \leq \text{position} \leq \text{limit} \leq \text{capacity}$

A newly-created buffer always has a position of zero and a mark that is undefined. The initial limit may be zero, or it may be some other value that depends upon the type of the buffer and the manner in which it is constructed. The initial content of a buffer is, in general, undefined.

Clearing, flipping, and rewinding:

In addition to methods for accessing the position, limit, and capacity values and for marking and resetting, this class also defines the following operations upon buffers:

clear() (p. 858) makes a buffer ready for a new sequence of channel-read or relative put operations: It sets the limit to the capacity and the position to zero.

flip() (p. 858) makes a buffer ready for a new sequence of channel-write or relative get operations: It sets the limit to the current position and then sets the position to zero.

rewind() (p. 861) makes a buffer ready for re-reading the data that it already contains: It leaves the limit unchanged and sets the position to zero.

Read-only buffers:

Every buffer is readable, but not every buffer is writable. The mutation methods of each buffer

class are specified as optional operations that will throw a **ReadOnlyBufferException** (p. 2966) when invoked upon a read-only buffer. A read-only buffer does not allow its content to be changed, but its mark, position, and limit values are mutable. Whether or not a buffer is read-only may be determined by invoking its `isReadOnly` method.

Thread safety:

Buffers are not safe for use by multiple concurrent threads. If a buffer is to be used by more than one thread then access to the buffer should be controlled by appropriate synchronization.

Invocation chaining:

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained; for example, the sequence of statements

```
b.flip(); b.position(23); b.limit(42);
```

can be replaced by the single, more compact statement `b.flip().position(23).limit(42);`

6.158.2 Constructor & Destructor Documentation

6.158.2.1 `decaf::nio::Buffer::Buffer (int capacity)`

6.158.2.2 `decaf::nio::Buffer::Buffer (const Buffer & other)`

6.158.2.3 `virtual decaf::nio::Buffer::~~Buffer ()` [inline, virtual]

6.158.3 Member Function Documentation

6.158.3.1 `virtual int decaf::nio::Buffer::capacity () const` [inline, virtual]

Returns

this buffer's capacity.

6.158.3.2 `virtual Buffer& decaf::nio::Buffer::clear ()` [virtual]

Clears this buffer.

The position is set to zero, the limit is set to the capacity, and the mark is discarded.

Invoke this method before using a sequence of channel-read or put operations to fill this buffer. For example:

```
buf.clear(); // Prepare buffer for reading in.read(buf); // Read data
```

This method does not actually erase the data in the buffer, but it is named as if it did because it will most often be used in situations in which that might as well be the case.

Returns

a reference to this buffer.

6.158.3.3 `virtual Buffer& decaf::nio::Buffer::flip ()` [virtual]

Flips this buffer.

The limit is set to the current position and then the position is set to zero. If the mark is defined then it is discarded.

After a sequence of channel-read or put operations, invoke this method to prepare for a sequence of channel-write or relative get operations. For example:

```
buf.put(magic); // Prepend header
in.read(buf); // Read data into rest of buffer
buf.flip(); // Flip buffer
out.write(buf); // Write header + data to channel
```

This method is often used in conjunction with the compact method when transferring data from one place to another.

Returns

a reference to this buffer.

6.158.3.4 `virtual bool decaf::nio::Buffer::hasRemaining () const [inline, virtual]`

Tells whether there are any elements between the current position and the limit.

Returns

true if, and only if, there is at least one element remaining in this buffer.

6.158.3.5 `virtual bool decaf::nio::Buffer::isReadOnly () const [pure virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

6.158.3.6 `virtual int decaf::nio::Buffer::limit () const [inline, virtual]`

Returns

this buffers Limit

6.158.3.7 `virtual Buffer& decaf::nio::Buffer::limit (int newLimit) throw (lang::exceptions::IllegalArgumentException) [virtual]`

Sets this buffer's limit.

If the position is larger than the new limit then it is set to the new limit. If the mark is defined and larger than the new limit then it is discarded.

Parameters

newLimit The new limit value; must be no larger than this buffer's capacity.

Returns

A reference to This buffer

Exceptions

IllegalArgumentException if preconditions on the new pos don't hold.

6.158.3.8 virtual Buffer& decaf::nio::Buffer::mark () [virtual]

Sets this buffer's mark at its position.

Returns

a reference to this buffer.

6.158.3.9 virtual int decaf::nio::Buffer::position () const [inline, virtual]

Returns

the current position in the buffer

6.158.3.10 virtual Buffer& decaf::nio::Buffer::position (int *newPosition*) throw (lang::exceptions::IllegalArgumentException) [virtual]

Sets this buffer's position.

If the mark is defined and larger than the new position then it is discarded.

Parameters

newPosition The new position in the buffer to set.

Returns

a reference to This buffer.

Exceptions

IllegalArgumentException if preconditions on the new pos don't hold.

6.158.3.11 virtual int decaf::nio::Buffer::remaining () const [inline, virtual]

Returns the number of elements between the current position and the limit.

Returns

The number of elements remaining in this buffer

6.158.3.12 virtual Buffer& decaf::nio::Buffer::reset () throw (InvalidMarkException) [virtual]

Resets this buffer's position to the previously-marked position.

Returns

a reference to this buffer.

Exceptions

InvalidMarkException (p. 1997) - If the mark has not been set

6.158.3.13 virtual Buffer& decaf::nio::Buffer::rewind () [virtual]

Rewinds this buffer.

The position is set to zero and the mark is discarded.

Invoke this method before a sequence of channel-write or get operations, assuming that the limit has already been set appropriately. For example:

```
out.write(buf); // Write remaining data buf.rewind(); // Rewind buffer buf.get(array); // Copy data into array
```

Returns

a reference to this buffer.

6.158.4 Field Documentation

6.158.4.1 int decaf::nio::Buffer::_capacity [protected]

6.158.4.2 int decaf::nio::Buffer::_limit [protected]

6.158.4.3 int decaf::nio::Buffer::_mark [protected]

6.158.4.4 bool decaf::nio::Buffer::_markSet [protected]

6.158.4.5 int decaf::nio::Buffer::_position [mutable, protected]

The documentation for this class was generated from the following file:

- src/main/decaf/nio/Buffer.h

6.159 decaf::io::BufferedInputStream Class Reference

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

```
#include <src/main/decaf/io/BufferedInputStream.h>
```

Inheritance diagram for decaf::io::BufferedInputStream:

Public Member Functions

- **BufferedInputStream** (InputStream *stream, bool own=false)

Constructor.

- **BufferedInputStream** (**InputStream** *stream, int bufferSize, bool own=false) throw (lang::exceptions::IllegalArgumentException)

Constructor.

- virtual ~**BufferedInputStream** ()
- virtual int **available** () const throw (decaf::io::IOException)

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

- virtual void **close** () throw (decaf::io::IOException)

*Closes the **InputStream** (p. 1909) freeing any resources that might have been acquired during the lifetime of this stream.*

The default implementation of this method does nothing.

- virtual long long **skip** (long long num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Skips over and discards n bytes of data from this input stream.

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

Parameters

num *The number of bytes to skip.*

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

- virtual void **mark** (int readLimit)

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit *The max bytes read before marked position is invalid.*

- virtual void **reset** () throw (decaf::io::IOException)

Repositions this stream to the position at the time the mark method was last called on this input stream.

*If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2003) might be thrown. * If such an **IOException** (p. 2003) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 2003). * If an **IOException** (p. 2003) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 2003).*

Exceptions

***IOException** (p. 2003) if an I/O error occurs.*

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Protected Member Functions

- virtual int **doReadByte** () throw (decaf::io::IOException)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

6.159.1 Detailed Description

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

6.159.2 Constructor & Destructor Documentation

6.159.2.1 decaf::io::BufferedInputStream::BufferedInputStream (InputStream * stream, bool own = false)

Constructor.

Parameters

stream The target input stream to buffer.

own Indicates if we own the stream object, defaults to false.

6.159.2.2 `decaf::io::BufferedInputStream::BufferedInputStream (InputStream * stream, int bufferSize, bool own = false) throw (lang::exceptions::IllegalArgumentException)`

Constructor.

Parameters

stream The target input stream to buffer.
bufferSize The size in bytes to allocate for the internal buffer.
own Indicates if we own the stream object, defaults to false.

Exceptions

IllegalArgumentException is the size is zero or negative.

6.159.2.3 `virtual decaf::io::BufferedInputStream::~~BufferedInputStream ()`
[virtual]

6.159.3 Member Function Documentation

6.159.3.1 `virtual int decaf::io::BufferedInputStream::available () const throw (decaf::io::IOException)` [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented from [decaf::io::FilterInputStream](#) (p. 1773).

6.159.3.2 `virtual void decaf::io::BufferedInputStream::close () throw (decaf::io::IOException)` [virtual]

Closes the [InputStream](#) (p. 1909) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from [decaf::io::FilterInputStream](#) (p. 1774).

6.159.3.3 `virtual int decaf::io::BufferedInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p.1774).

6.159.3.4 `virtual int decaf::io::BufferedInputStream::doReadByte () throw (decaf::io::IOException)` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p.1774).

6.159.3.5 `virtual void decaf::io::BufferedInputStream::mark (int readLimit)` [virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit The max bytes read before marked position is invalid.

Reimplemented from `decaf::io::FilterInputStream` (p.1775).

6.159.3.6 `virtual bool decaf::io::BufferedInputStream::markSupported () const` [inline, virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented from `decaf::io::FilterInputStream` (p.1775).

6.159.3.7 `virtual void decaf::io::BufferedInputStream::reset () throw (decaf::io::IOException)` [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method `markSupported` returns true, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since `mark` was last called is larger than the argument to `mark` at that last call, then an **IOException** (p. 2003) might be thrown. * If such an **IOException** (p. 2003) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to `mark` (or since the start of the file, if `mark` has not been called) will be resupplied to subsequent callers of the `read` method, followed by any bytes that otherwise would have been the next input data as of the time of the call to `reset`.

If the method `markSupported` returns false, then: * The call to `reset` may throw an **IOException** (p. 2003). * If an **IOException** (p. 2003) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the `read` method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2003).

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p. 1775).

6.159.3.8 virtual long long decaf::io::BufferedInputStream::skip
 (long long *num*) throw (decaf::io::IOException,
 decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Skips over and discards *n* bytes of data from this input stream.

The `skip` method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The `skip` method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::FilterInputStream** (p. 1776).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/BufferedInputStream.h`

6.160 decaf::io::BufferedOutputStream Class Reference

Wrapper around another output stream that buffers output before writing to the target output stream.

```
#include <src/main/decaf/io/BufferedOutputStream.h>
```

Inheritance diagram for decaf::io::BufferedOutputStream:

Public Member Functions

- **BufferedOutputStream** (**OutputStream** *stream, bool **own**=false)
Constructor.
- **BufferedOutputStream** (**OutputStream** *stream, int bufferSize, bool **own**=false) throw (decaf::lang::exceptions::IllegalArgumentException)
Constructor.
- virtual ~**BufferedOutputStream** ()
- virtual void **flush** () throw (decaf::io::IOException)
inheritDoc
- virtual void **doWriteByte** (unsigned char c) throw (decaf::io::IOException)
- virtual void **doWriteArray** (const unsigned char *buffer, int size) throw (decaf::io::IOException)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

6.160.1 Detailed Description

Wrapper around another output stream that buffers output before writing to the target output stream.

6.160.2 Constructor & Destructor Documentation

6.160.2.1 decaf::io::BufferedOutputStream::BufferedOutputStream (**OutputStream** * *stream*, bool *own* = *false*)

Constructor.

Parameters

stream The target output stream.

own Indicates if this class owns the stream pointer.

6.160.2.2 `decaf::io::BufferedOutputStream::BufferedOutputStream (OutputStream * stream, int bufferSize, bool own = false) throw (decaf::lang::exceptions::IllegalArgumentException)`

Constructor.

Parameters

stream The target output stream.

bufferSize The size for the internal buffer.

own Indicates if this class owns the stream pointer.

Exceptions

IllegalArgumentException if the *bufferSize* given is negative.

6.160.2.3 `virtual decaf::io::BufferedOutputStream::~~BufferedOutputStream ()`
[virtual]

6.160.3 Member Function Documentation

6.160.3.1 `virtual void decaf::io::BufferedOutputStream::doWriteArray (const unsigned char * buffer, int size) throw (decaf::io::IOException)`
[protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1779).

6.160.3.2 `virtual void decaf::io::BufferedOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1779).

6.160.3.3 `virtual void decaf::io::BufferedOutputStream::doWriteByte (unsigned char c) throw (decaf::io::IOException)` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1779).

6.160.3.4 `virtual void decaf::io::BufferedOutputStream::flush () throw (decaf::io::IOException)` [virtual]

inheritDoc}

Reimplemented from `decaf::io::FilterOutputStream` (p. 1779).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/BufferedOutputStream.h`

6.161 decaf::internal::nio::BufferFactory Class Reference

Factory class used by static methods in the **decaf::nio** (p.132) package to create the various default version of the NIO interfaces.

```
#include <src/main/decaf/internal/nio/BufferFactory.h>
```

Public Member Functions

- virtual **~BufferFactory** ()

Static Public Member Functions

- static **decaf::nio::ByteBuffer** * **createByteBuffer** (int capacity) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::ByteBuffer** * **createByteBuffer** (unsigned char *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Wraps the passed buffer with a new ByteBuffer.

- static **decaf::nio::ByteBuffer** * **createByteBuffer** (std::vector< unsigned char > &buffer)

Wraps the passed STL Byte Vector in a ByteBuffer.

- static **decaf::nio::CharBuffer** * **createCharBuffer** (int capacity) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::CharBuffer** * **createCharBuffer** (char *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Wraps the passed buffer with a new CharBuffer.

- static **decaf::nio::CharBuffer** * **createCharBuffer** (std::vector< char > &buffer)

Wraps the passed STL Byte Vector in a CharBuffer.

- static **decaf::nio::DoubleBuffer** * **createDoubleBuffer** (int capacity) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::DoubleBuffer** * **createDoubleBuffer** (double *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Wraps the passed buffer with a new DoubleBuffer.

- static **decaf::nio::DoubleBuffer** * **createDoubleBuffer** (std::vector< double > &buffer)
Wraps the passed STL Double Vector in a DoubleBuffer.
- static **decaf::nio::FloatBuffer** * **createFloatBuffer** (int capacity) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::FloatBuffer** * **createFloatBuffer** (float *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Wraps the passed buffer with a new FloatBuffer.
- static **decaf::nio::FloatBuffer** * **createFloatBuffer** (std::vector< float > &buffer)
Wraps the passed STL Float Vector in a FloatBuffer.
- static **decaf::nio::LongBuffer** * **createLongBuffer** (int capacity) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::LongBuffer** * **createLongBuffer** (long long *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Wraps the passed buffer with a new LongBuffer.
- static **decaf::nio::LongBuffer** * **createLongBuffer** (std::vector< long long > &buffer)
Wraps the passed STL Long Vector in a LongBuffer.
- static **decaf::nio::IntBuffer** * **createIntBuffer** (int capacity) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::IntBuffer** * **createIntBuffer** (int *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Wraps the passed buffer with a new IntBuffer.
- static **decaf::nio::IntBuffer** * **createIntBuffer** (std::vector< int > &buffer)
Wraps the passed STL int Vector in a IntBuffer.
- static **decaf::nio::ShortBuffer** * **createShortBuffer** (int capacity) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::ShortBuffer** * **createShortBuffer** (short *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Wraps the passed buffer with a new ShortBuffer.

- static **decaf::nio::ShortBuffer * createShortBuffer** (std::vector< short > &buffer)
Wraps the passed STL Short Vector in a ShortBuffer.

6.161.1 Detailed Description

Factory class used by static methods in the **decaf::nio** (p.132) package to create the various default version of the NIO interfaces.

Since

1.0

6.161.2 Constructor & Destructor Documentation

- 6.161.2.1** virtual **decaf::internal::nio::BufferFactory::~~BufferFactory** () [inline, virtual]

6.161.3 Member Function Documentation

- 6.161.3.1** static **decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer** (int *capacity*) throw (**decaf::lang::exceptions::IndexOutOfBoundsException**) [static]

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity The internal buffer's capacity.

Returns

a newly allocated ByteBuffer which the caller owns.

Exceptions

IndexOutOfBoundsException if the capacity specified is negative.

- 6.161.3.2** static **decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer** (unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (**decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IndexOutOfBoundsException**) [static]

Wraps the passed buffer with a new ByteBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The array that will back the new buffer.
size The size of the specified buffer.
offset The offset of the subarray to be used.
length The length of the subarray to be used.

Returns

a new ByteBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

NullPointerException if the buffer given is Null.
IndexOutOfBoundsException if the capacity specified is negative.

6.161.3.3 static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (std::vector< unsigned char > & *buffer*) [static]

Wraps the passed STL Byte Vector in a ByteBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns

a new ByteBuffer that is backed by buffer, caller owns.

6.161.3.4 static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (int *capacity*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [static]

Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity The internal buffer's capacity.

Returns

a newly allocated CharBuffer which the caller owns.

Exceptions

IndexOutOfBoundsException if the capacity specified is negative.

6.161.3.5 `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (char * buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [static]`

Wraps the passed buffer with a new CharBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The array that will back the new buffer.

size The size of the specified buffer.

offset The offset of the subarray to be used.

length The length of the subarray to be used.

Returns

a new CharBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

NullPointerException if the buffer given in Null.

IndexOutOfBoundsException if the capacity specified is negative.

6.161.3.6 `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (std::vector< char > & buffer) [static]`

Wraps the passed STL Byte Vector in a CharBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns

a new CharBuffer that is backed by buffer, caller owns.

6.161.3.7 `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (double * buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [static]`

Wraps the passed buffer with a new DoubleBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be offset, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The array that will back the new buffer.
size The size of the specified buffer.
offset The offset of the subarray to be used.
length The length of the subarray to be used.

Returns

a new `DoubleBuffer` that is backed by `buffer`, caller owns the returned pointer.

Exceptions

NullPointerException if the buffer given is Null.
IndexOutOfBoundsException if the capacity specified is negative.

6.161.3.8 `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (std::vector< double > & buffer) [static]`

Wraps the passed STL Double Vector in a `DoubleBuffer`.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new `DoubleBuffer` that is backed by `buffer`, caller owns.

6.161.3.9 `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (int capacity) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [static]`

Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity The internal buffer's capacity.

Returns

a newly allocated DoubleBuffer which the caller owns.

Exceptions

IndexOutOfBoundsException if the capacity specified is negative.

```
6.161.3.10 static decaf::nio::FloatBuffer* de-
caf::internal::nio::BufferFactory::createFloatBuffer ( int
capacity ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException
) [static]
```

Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity The internal buffer's capacity.

Returns

a newly allocated FloatBuffer which the caller owns.

Exceptions

IndexOutOfBoundsException if the capacity specified is negative.

```
6.161.3.11 static decaf::nio::FloatBuffer* de-
caf::internal::nio::BufferFactory::createFloatBuffer (
float * buffer, int size, int offset, int length )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException ) [static]
```

Wraps the passed buffer with a new FloatBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The array that will back the new buffer.

size The size of the specified buffer.

offset The offset of the subarray to be used.

length The length of the subarray to be used.

Returns

a new FloatBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

NullPointerException if the buffer given in Null.

IndexOutOfBoundsException if the capacity specified is negative.

6.161.3.12 `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (std::vector< float > & buffer) [static]`

Wraps the passed STL Float Vector in a FloatBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new FloatBuffer that is backed by *buffer*, caller owns.

6.161.3.13 `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (int capacity) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [static]`

Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity The internal buffer's capacity.

Returns

a newly allocated IntBuffer which the caller owns.

Exceptions

IndexOutOfBoundsException if the capacity specified is negative.

6.161.3.14 `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (std::vector< int > & buffer) [static]`

Wraps the passed STL int Vector in a IntBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new `IntBuffer` that is backed by `buffer`, caller owns.

```
6.161.3.15  static decaf::nio::IntBuffer* de-
             caf::internal::nio::BufferFactory::createIntBuffer ( int
             * buffer, int size, int offset, int length )
             throw ( decaf::lang::exceptions::NullPointerException,
             decaf::lang::exceptions::IndexOutOfBoundsException ) [static]
```

Wraps the passed `buffer` with a new `IntBuffer`.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The array that will back the new buffer.

size The size of the specified buffer.

offset The offset of the subarray to be used.

length The length of the subarray to be used.

Returns

a new `IntBuffer` that is backed by `buffer`, caller owns the returned pointer.

Exceptions

NullPointerException if the `buffer` given is `Null`.

IndexOutOfBoundsException if the capacity specified is negative.

```
6.161.3.16  static decaf::nio::LongBuffer* de-
             caf::internal::nio::BufferFactory::createLongBuffer (
             std::vector< long long > & buffer ) [static]
```

Wraps the passed STL Long Vector in a `LongBuffer`.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new `LongBuffer` that is backed by `buffer`, caller owns.

6.161.3.17 static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (int *capacity*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [static]

Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity - the internal buffer's capacity.

Returns

a newly allocated DoubleBuffer which the caller owns.

Exceptions

IndexOutOfBoundsException if the capacity specified is negative.

6.161.3.18 static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (long * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [static]

Wraps the passed buffer with a new LongBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The array that will back the new buffer.

size The size of the specified buffer.

offset The offset of the subarray to be used.

length The length of the subarray to be used.

Returns

a new LongBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

NullPointerException if the buffer given in Null.

IndexOutOfBoundsException if the capacity specified is negative.

6.161.3.19 static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (int *capacity*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [static]

Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity The internal buffer's capacity.

Returns

a newly allocated ShortBuffer which the caller owns.

Exceptions

IndexOutOfBoundsException if the capacity specified is negative.

```
6.161.3.20 static decaf::nio::ShortBuffer* de-
caf::internal::nio::BufferFactory::createShortBuffer (
short * buffer, int size, int offset, int length )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException ) [static]
```

Wraps the passed buffer with a new ShortBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The array that will back the new buffer.

size The size of the specified buffer.

offset The offset of the subarray to be used.

length The length of the subarray to be used.

Returns

a new ShortBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

NullPointerException if the buffer given is Null.

IndexOutOfBoundsException if the capacity specified is negative.

```
6.161.3.21 static decaf::nio::ShortBuffer* de-
caf::internal::nio::BufferFactory::createShortBuffer (
std::vector< short > & buffer ) [static]
```

Wraps the passed STL Short Vector in a ShortBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns

a new DoubleBuffer that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**BufferFactory.h**

6.162 decaf::nio::BufferOverflowException Class Reference

```
#include <src/main/decaf/nio/BufferOverflowException.h>
```

Inheritance diagram for decaf::nio::BufferOverflowException:

Public Member Functions

- **BufferOverflowException** () throw ()
Default Constructor.
- **BufferOverflowException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **BufferOverflowException** (const BufferOverflowException &ex) throw ()
Copy Constructor.
- **BufferOverflowException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BufferOverflowException** (const std::exception *cause) throw ()
Constructor.
- **BufferOverflowException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **BufferOverflowException** * clone () const
Clones this exception.
- virtual ~**BufferOverflowException** () throw ()

6.162.1 Constructor & Destructor Documentation

6.162.1.1 decaf::nio::BufferOverflowException::BufferOverflowException () throw () [inline]

Default Constructor.

6.162.1.2 `decaf::nio::BufferOverflowException::BufferOverflowException (const lang::Exception & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy

6.162.1.3 `decaf::nio::BufferOverflowException::BufferOverflowException (const BufferOverflowException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.162.1.4 `decaf::nio::BufferOverflowException::BufferOverflowException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.162.1.5 `decaf::nio::BufferOverflowException::BufferOverflowException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.162.1.6 `decaf::nio::BufferOverflowException::BufferOverflowException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.162.1.7 virtual decaf::nio::BufferOverflowException::~~BufferOverflowException () throw () [inline, virtual]

6.162.2 Member Function Documentation

6.162.2.1 virtual BufferOverflowException* decaf::nio::BufferOverflowException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/BufferOverflowException.h

6.163 decaf::nio::BufferUnderflowException Class Reference

```
#include <src/main/decaf/nio/BufferUnderflowException.h>
```

Inheritance diagram for decaf::nio::BufferUnderflowException:

Public Member Functions

- **BufferUnderflowException** () throw ()
Default Constructor.
- **BufferUnderflowException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **BufferUnderflowException** (const BufferUnderflowException &ex) throw ()
Copy Constructor.
- **BufferUnderflowException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BufferUnderflowException** (const std::exception *cause) throw ()

Constructor.

- **BufferUnderflowException** (const char *file, const int lineNumber, const char *msg,...)
throw ()

Constructor.

- virtual **BufferUnderflowException** * clone () const

Clones this exception.

- virtual ~**BufferUnderflowException** () throw ()

6.163.1 Constructor & Destructor Documentation

6.163.1.1 decaf::nio::BufferUnderflowException::BufferUnderflowException () throw () [inline]

Default Constructor.

6.163.1.2 decaf::nio::BufferUnderflowException::BufferUnderflowException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.163.1.3 decaf::nio::BufferUnderflowException::BufferUnderflowException (const BufferUnderflowException & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.163.1.4 decaf::nio::BufferUnderflowException::BufferUnderflowException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.163.1.5 `decaf::nio::BufferUnderflowException::BufferUnderflowException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.163.1.6 `decaf::nio::BufferUnderflowException::BufferUnderflowException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.163.1.7 `virtual decaf::nio::BufferUnderflowException::~~BufferUnderflowException () throw () [inline, virtual]`

6.163.2 Member Function Documentation

6.163.2.1 `virtual BufferUnderflowException* decaf::nio::BufferUnderflowException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from `decaf::lang::Exception` (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/BufferUnderflowException.h`

6.164 decaf::lang::Byte Class Reference

```
#include <src/main/decaf/lang/Byte.h>
```

Inheritance diagram for `decaf::lang::Byte`:

Public Member Functions

- **Byte** (unsigned char value)
- **Byte** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Byte** ()
- virtual int **compareTo** (const **Byte** &c) const
*Compares this **Byte** (p. 884) instance with another.*
- virtual bool **operator==** (const **Byte** &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Byte** &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const unsigned char &c) const
*Compares this **Byte** (p. 884) instance with a char type.*
- virtual bool **operator==** (const unsigned char &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const unsigned char &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const **Byte** &c) const
- bool **equals** (const unsigned char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static std::string **toString** (unsigned char value)
- static **Byte decode** (const std::string &value) throw (exceptions::NumberFormatException)
*Decodes a **String** (p. 3427) into a **Byte** (p. 884).*
- static unsigned char **parseByte** (const std::string &s, int radix) throw (exceptions::NumberFormatException)
Parses the string argument as a signed unsigned char in the radix specified by the second argument.
- static unsigned char **parseByte** (const std::string &s) throw (exceptions::NumberFormatException)
Parses the string argument as a signed decimal unsigned char.
- static **Byte valueOf** (unsigned char value)
*Returns a **Character** (p. 1019) instance representing the specified char value.*
- static **Byte valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Byte** (p. 884) object holding the value given by the specified std::string.*
- static **Byte valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Byte** (p. 884) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const unsigned char **MIN_VALUE** = 0x7F
The minimum value that a unsigned char can take on.
- static const unsigned char **MAX_VALUE** = 0x80
The maximum value that a unsigned char can take on.
- static const int **SIZE** = 8
The size of the primitive character in bits.

6.164.1 Constructor & Destructor Documentation

6.164.1.1 decaf::lang::Byte::Byte (unsigned char value)

Parameters

value - the primitive value to wrap

6.164.1.2 `decaf::lang::Byte::Byte (const std::string & value) throw (exceptions::NumberFormatException)`

Parameters

value - the string to convert to an unsigned char

Exceptions

NumberFormatException

6.164.1.3 `virtual decaf::lang::Byte::~~Byte () [inline, virtual]`

6.164.2 Member Function Documentation

6.164.2.1 `virtual unsigned char decaf::lang::Byte::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

6.164.2.2 `virtual int decaf::lang::Byte::compareTo (const unsigned char & c) const [inline, virtual]`

Compares this **Byte** (p. 884) instance with a char type.

Parameters

c - the char instance to be compared

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< unsigned char >` (p. 1126).

6.164.2.3 `virtual int decaf::lang::Byte::compareTo (const Byte & c) const [inline, virtual]`

Compares this **Byte** (p. 884) instance with another.

Parameters

c - the **Byte** (p. 884) instance to be compared

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.164.2.4 `static Byte decaf::lang::Byte::decode (const std::string & value) throw (exceptions::NumberFormatException) [static]`

Decodes a **String** (p. 3427) into a **Byte** (p. 884).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Byte::parseByte** (p. 891) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified **String** (p. 3427) is the minus sign. No whitespace characters are permitted in the string.

Parameters

value - The string to decode

Returns

a **Byte** (p. 884) object containing the decoded value

Exceptions

NumberFormatException if the string is not formatted correctly.

6.164.2.5 `virtual double decaf::lang::Byte::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.164.2.6 `bool decaf::lang::Byte::equals (const unsigned char & c) const [inline, virtual]`

Returns

true if the two Bytes have the same value.

Implements **decaf::lang::Comparable< unsigned char >** (p. 1126).

6.164.2.7 `bool decaf::lang::Byte::equals (const Byte & c) const [inline]`

Returns

true if the two **Byte** (p. 884) Objects have the same value.

6.164.2.8 `virtual float decaf::lang::Byte::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.164.2.9 `virtual int decaf::lang::Byte::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.164.2.10 `virtual long long decaf::lang::Byte::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long long the value of the receiver.

6.164.2.11 `virtual bool decaf::lang::Byte::operator< (const unsigned char & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< unsigned char >` (p.1127).

6.164.2.12 `virtual bool decaf::lang::Byte::operator< (const Byte & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.164.2.13 `virtual bool decaf::lang::Byte::operator==(const Byte & c) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

c - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.164.2.14 `virtual bool decaf::lang::Byte::operator==(const unsigned char & c) const` [inline, virtual]

Compares equality between this object and the one passed.

Parameters

c - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< unsigned char >` (p.1127).

6.164.2.15 `static unsigned char decaf::lang::Byte::parseByte (const std::string & s) throw (exceptions::NumberFormatException)` [static]

Parses the string argument as a signed decimal unsigned char.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting unsigned char value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseByte(const std::string, int)` method.

Parameters

s - `String` (p.3427) to convert to a unsigned char

Returns

the converted unsigned char value

Exceptions

NumberFormatException if the string is not a unsigned char.

6.164.2.16 `static unsigned char decaf::lang::Byte::parseByte (const std::string & s, int radix) throw (exceptions::NumberFormatException) [static]`

Parses the string argument as a signed unsigned char in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 1022) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type `NumberFormatException` is thrown if any of the following situations occurs:
 * The first argument is null or is a string of length zero. * The radix is either smaller than **Character.MIN_RADIX** (p. 1026) or larger than **Character::MAX_RADIX** (p. 1026). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type unsigned char.

Parameters

s - the **String** (p. 3427) containing the unsigned char to be parsed

radix - the radix to be used while parsing s

Returns

the unsigned char represented by the string argument in the specified radix.

Exceptions

NumberFormatException - If **String** (p. 3427) does not contain a parsable unsigned char.

6.164.2.17 `virtual short decaf::lang::Byte::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

6.164.2.18 `static std::string decaf::lang::Byte::toString (unsigned char value) [static]`

Returns

a string representing the primitive value as Base 10

6.164.2.19 `std::string decaf::lang::Byte::toString () const`

Returns

this **Byte** (p. 884) Object as a **String** (p. 3427) Representation

6.164.2.20 `static Byte decaf::lang::Byte::valueOf (unsigned char value)`
 `[inline, static]`

Returns a **Character** (p.1019) instance representing the specified char value.

Parameters

value - the primitive char to wrap.

Returns

a new **Character** (p.1019) instance that wraps this value.

6.164.2.21 `static Byte decaf::lang::Byte::valueOf (const std::string & value, int`
 `radix) throw (exceptions::NumberFormatException) [static]`

Returns a **Byte** (p.884) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed unsigned char in the radix specified by the second argument, exactly as if the argument were given to the `parseByte(std::string, int)` method. The result is a **Byte** (p.884) object that represents the unsigned char value specified by the string.

Parameters

value - std::string to parse as base (radix)

radix - base of the string to parse.

Returns

new **Byte** (p.884) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a valid unsigned char.

6.164.2.22 `static Byte decaf::lang::Byte::valueOf (const std::string & value)`
 `throw (exceptions::NumberFormatException) [static]`

Returns a **Byte** (p.884) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal unsigned char, exactly as if the argument were given to the `parseByte(std::string)` method. The result is a **Byte** (p.884) object that represents the unsigned char value specified by the string.

Parameters

value - std::string to parse as base 10

Returns

new **Byte** (p.884) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a decimal unsigned char.

6.164.3 Field Documentation

6.164.3.1 `const unsigned char decaf::lang::Byte::MAX_VALUE = 0x80` [static]

The maximum value that a unsigned char can take on.

6.164.3.2 `const unsigned char decaf::lang::Byte::MIN_VALUE = 0x7F` [static]

The minimum value that a unsigned char can take on.

6.164.3.3 `const int decaf::lang::Byte::SIZE = 8` [static]

The size of the primitive charactor in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Byte.h`

6.165 decaf::internal::util::ByteArrayAdapter Class Reference

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

```
#include <src/main/decaf/internal/util/ByteArrayAdapter.h>
```

Data Structures

- `union Array`

Public Member Functions

- **ByteArrayAdapter** (int size) throw (decaf::lang::exceptions::IllegalArgumentException)
Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.
- **ByteArrayAdapter** (unsigned char *array, int size, bool own=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (char *array, int size, bool own=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (double *array, int size, bool own=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte array object that wraps the given array.

- **ByteArrayAdapter** (float *array, int size, bool own=false)
throw (decaf::lang::exceptions::NullPointerException, de-
caf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte array object that wraps the given array.

- **ByteArrayAdapter** (long long *array, int size, bool
own=false) throw (decaf::lang::exceptions::NullPointerException, de-
caf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte array object that wraps the given array.

- **ByteArrayAdapter** (int *array, int size, bool own=false)
throw (decaf::lang::exceptions::NullPointerException, de-
caf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte array object that wraps the given array.

- **ByteArrayAdapter** (short *array, int size, bool own=false)
throw (decaf::lang::exceptions::NullPointerException, de-
caf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte array object that wraps the given array.

- virtual ~**ByteArrayAdapter** ()

- virtual int **getCapacity** () const

Gets the size of the underlying array.

- virtual int **getCharCapacity** () const

Gets the size of the underlying array as if it contains chars.

- virtual int **getDoubleCapacity** () const

Gets the size of the underlying array as if it contains doubles.

- virtual int **getFloatCapacity** () const

Gets the size of the underlying array as if it contains doubles.

- virtual int **getLongCapacity** () const

Gets the size of the underlying array as if it contains doubles.

- virtual int **getIntCapacity** () const

Gets the size of the underlying array as if it contains ints.

- virtual int **getShortCapacity** () const

Gets the size of the underlying array as if it contains shorts.

- virtual unsigned char * **getByteArray** ()

Gets the pointer to the array we are wrapping.

- virtual char * **getCharArray** ()

Gets the pointer to the array we are wrapping.

- virtual short * **getShortArray** ()

Gets the pointer to the array we are wrapping.

- virtual int * **getIntArray** ()

Gets the pointer to the array we are wrapping.

- virtual long long * **getLongArray** ()

Gets the pointer to the array we are wrapping.

- virtual double * **getDoubleArray** ()

Gets the pointer to the array we are wrapping.

- virtual float * **getFloatArray** ()

Gets the pointer to the array we are wrapping.

- virtual void **read** (unsigned char *buffer, int size, int offset, int length) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException)

Reads from the Byte array starting at the specified offset and reading the specified length.

- virtual void **write** (unsigned char *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException)

Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array.

- virtual void **resize** (int size) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InvalidStateException)

Resizes the underlying array to the new given size, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.

- virtual void **clear** () throw ()

Clear all data from that Array, setting the underlying bytes to zero.

- unsigned char & **operator[]** (int index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

*Allows the **ByteArrayAdapter** (p. 893) to be indexed as a standard array.*

- const unsigned char & **operator[]** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

- virtual unsigned char **get** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual char **getChar** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Reads one byte at the given index and returns it.

- virtual double **getDouble** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Reads eight bytes at the given index and returns it.

- virtual double **getDoubleAt** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given byte index and returns it.
- virtual float **getFloat** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.
- virtual float **getFloatAt** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given byte index and returns it.
- virtual long long **getLong** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given index and returns it.
- virtual long long **getLongAt** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given byte index and returns it.
- virtual int **getInt** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.
- virtual int **getIntAt** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given byte index and returns it.
- virtual short **getShort** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads two bytes at the given index and returns it.
- virtual short **getShortAt** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads two bytes at the given byte index and returns it.
- virtual **ByteArrayAdapter** & **put** (int index, unsigned char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Writes the given byte into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putChar** (int index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Writes one byte containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putDouble** (int index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putDoubleAt** (int index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Writes eight bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putFloat** (int index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putFloatAt** (int index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Writes four bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putLong** (int index, long long value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putLongAt** (int index, long long value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Writes eight bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putInt** (int index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putIntAt** (int index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Writes four bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putShort** (int index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Writes two bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putShortAt** (int index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Writes two bytes containing the given value, into this buffer at the given byte index.

6.165.1 Detailed Description

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data. All the array types are mapped down to a byte array and methods are supplied for accessing the data in any of the primitive type forms.

Methods in this class that do not return a specific value return a reference to this object so that calls can be chained.

Since

1.0

6.165.2 Constructor & Destructor Documentation

6.165.2.1 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (int *size*) throw (decaf::lang::exceptions::IllegalArgumentException)

Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

size The size of the array, this is the limit we read and write to.

Exceptions

IllegalArgumentException if size is negative.

6.165.2.2 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (unsigned char * *array*, int *size*, bool *own* = *false*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The physical array to wrap.

size The size of the array, this is the limit we read and write to.

own Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

IndexOutOfBoundsException if the size is negative.

6.165.2.3 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (char * *array*, int *size*, bool *own* = *false*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The physical array to wrap.

size The size of the array, this is the limit we read and write to.

own Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

IndexOutOfBoundsException if the size is negative.

6.165.2.4 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter`
 (`double * array`, `int size`, `bool own = false`)
 throw (`decaf::lang::exceptions::NullPointerException`,
`decaf::lang::exceptions::IndexOutOfBoundsException`)

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The physical array to wrap.

size The size of the array, this is the limit we read and write to.

own Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

IndexOutOfBoundsException if the size is negative.

6.165.2.5 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter`
 (`float * array`, `int size`, `bool own = false`)
 throw (`decaf::lang::exceptions::NullPointerException`,
`decaf::lang::exceptions::IndexOutOfBoundsException`)

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The physical array to wrap.

size The size of the array, this is the limit we read and write to.

own Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

IndexOutOfBoundsException if the size is negative.

6.165.2.6 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter`
 (`long long * array`, `int size`, `bool own = false`)
 throw (`decaf::lang::exceptions::NullPointerException`,
`decaf::lang::exceptions::IndexOutOfBoundsException`)

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The physical array to wrap.
size The size of the array, this is the limit we read and write to.
own Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL
IndexOutOfBoundsException if the size is negative.

6.165.2.7 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter
(int * array, int size, bool own = false)
throw (decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The physical array to wrap.
size The size of the array, this is the limit we read and write to.
own Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL
IndexOutOfBoundsException if the size is negative.

6.165.2.8 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter
(short * array, int size, bool own = false)
throw (decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The physical array to wrap.
size The size of the array, this is the limit we read and write to.
own Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL
IndexOutOfBoundsException if the size is negative.

6.165.2.9 `virtual decaf::internal::util::ByteArrayAdapter::~~ByteArrayAdapter ()`
[virtual]

6.165.3 Member Function Documentation

6.165.3.1 `virtual void decaf::internal::util::ByteArrayAdapter::clear () throw ()`
[virtual]

Clear all data from that Array, setting the underlying bytes to zero.

6.165.3.2 `virtual unsigned char decaf::internal::util::ByteArrayAdapter::get`
`(int index) const throw (de-`
`caf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Absolute get method.

Reads the byte at the given index.

Parameters

index The index in the Buffer where the byte is to be read.

Returns

the byte that is located at the given index.

Exceptions

IndexOutOfBoundsException If index is not smaller than the buffer's limit or is negative.

6.165.3.3 `virtual unsigned char* de-`
`caf::internal::util::ByteArrayAdapter::getByteArray ()`
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 893) objects that point to this array.

Returns

an unsigned char* pointer to the array this object wraps.

6.165.3.4 `virtual int decaf::internal::util::ByteArrayAdapter::getCapacity ()`
`const` [inline, virtual]

Gets the size of the underlying array.

Returns

the size the array.

6.165.3.5 `virtual char decaf::internal::util::ByteArrayAdapter::getChar (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Reads one byte at the given index and returns it.

Parameters

index The index in the Buffer where the byte is to be read.

Returns

the byte that is located at the given index.

Exceptions

IndexOutOfBoundsException If index is not smaller than the buffer's limit or is negative.

6.165.3.6 `virtual char* decaf::internal::util::ByteArrayAdapter::getCharArray ()`
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 893) objects that point to this array.

Returns

an char* pointer to the array this object wraps.

6.165.3.7 `virtual int decaf::internal::util::ByteArrayAdapter::getCharCapacity ()`
const [inline, virtual]

Gets the size of the underlying array as if it contains chars.

Returns

the size the array.

6.165.3.8 `virtual double decaf::internal::util::ByteArrayAdapter::getDouble (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Reads eight bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The index in the Buffer where the bytes are to be read.

Returns

the value at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.165.3.9 **virtual double* decaf::internal::util::ByteArrayAdapter::getDoubleArray**
 () [inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 893) objects that point to this array.

Returns

an double* pointer to the array this object wraps.

6.165.3.10 **virtual double decaf::internal::util::ByteArrayAdapter::getDoubleAt**
 (int *index*) const throw (de-
 caf::lang::exceptions::IndexOutOfBoundsException)
 [virtual]

Reads eight bytes at the given byte index and returns it.

Parameters

index The index in the Buffer where the bytes are to be read

Returns

the value at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.165.3.11 **virtual int decaf::internal::util::ByteArrayAdapter::getDoubleCapacity (**
) const [inline, virtual]

Gets the size of the underlying array as if it contains doubles.

Returns

the size the array.

6.165.3.12 `virtual float decaf::internal::util::ByteArrayAdapter::getFloat (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Reads four bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The index in the Buffer where the bytes are to be read.

Returns

the value at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.165.3.13 `virtual float* decaf::internal::util::ByteArrayAdapter::getFloatArray ()`
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 893) objects that point to this array.

Returns

an float* pointer to the array this object wraps.

6.165.3.14 `virtual float decaf::internal::util::ByteArrayAdapter::getFloatAt (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Reads four bytes at the given byte index and returns it.

Parameters

index The index in the Buffer where the bytes are to be read

Returns

the value at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.165.3.15 `virtual int decaf::internal::util::ByteArrayAdapter::getFloatCapacity () const [inline, virtual]`

Gets the size of the underlying array as if it contains doubles.

Returns

the size the array.

6.165.3.16 `virtual int decaf::internal::util::ByteArrayAdapter::getInt (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads four bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The index in the Buffer where the bytes are to be read.

Returns

the value at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.165.3.17 `virtual int* decaf::internal::util::ByteArrayAdapter::getIntArray () [inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 893) objects that point to this array.

Returns

an int* pointer to the array this object wraps.

6.165.3.18 `virtual int decaf::internal::util::ByteArrayAdapter::getIntAt (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads four bytes at the given byte index and returns it.

Parameters

index The index in the Buffer where the bytes are to be read

Returns

the value at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.165.3.19 `virtual int decaf::internal::util::ByteArrayAdapter::getIntCapacity ()
 const [inline, virtual]`

Gets the size of the underlying array as if it contains ints.

Returns

the size the array.

6.165.3.20 `virtual long long decaf::internal::util::ByteArrayAdapter::getLong
 (int index) const throw (de-
 ccaf::lang::exceptions::IndexOutOfBoundsException)
 [virtual]`

Reads eight bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The index in the Buffer where the bytes are to be read.

Returns

the value at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.165.3.21 `virtual long long* decaf::internal::util::ByteArrayAdapter::getLongArray
 () [inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 893) objects that point to this array.

Returns

an long long* pointer to the array this object wraps.

6.165.3.22 `virtual long long decaf::internal::util::ByteArrayAdapter::getLongAt (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads eight bytes at the given byte index and returns it.

Parameters

index The index in the Buffer where the bytes are to be read

Returns

the value at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.165.3.23 `virtual int decaf::internal::util::ByteArrayAdapter::getLongCapacity () const [inline, virtual]`

Gets the size of the underlying array as if it contains doubles.

Returns

the size the array.

6.165.3.24 `virtual short decaf::internal::util::ByteArrayAdapter::getShort (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads two bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The index in the Buffer where the bytes are to be read.

Returns

the value at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

```
6.165.3.25 virtual short* decaf::internal::util::ByteArrayAdapter::getShortArray (
) [inline, virtual]
```

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p.893) objects that point to this array.

Returns

an `short*` pointer to the array this object wraps.

```

6.165.3.26 virtual short decaf::internal::util::ByteArrayAdapter::getShortAt
( int index ) const throw ( de-
caf::lang::exceptions::IndexOutOfBoundsException )
[virtual]

```

Reads two bytes at the given byte index and returns it.

Parameters

index The index in the Buffer where the bytes are to be read

Returns

the value at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

```
6.165.3.27 virtual int decaf::internal::util::ByteArrayAdapter::getShortCapacity (
) const [inline, virtual]
```

Gets the size of the underlying array as if it contains shorts.

Returns

the size the array.

6.165.3.28 unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (int *index*) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Allows the **ByteArrayAdapter** (p. 893) to be indexed as a standard array.
calling the non constant version allows the user to change the value at index

Parameters

index The position in the array to access, if the value is negative or greater than the size of the underlying array an `IndexOutOfBoundsException` is thrown.

Exceptions

IndexOutOfBoundsException if the preconditions of index are not met.

6.165.3.29 `const unsigned char& decaf::internal::util::ByteArrayAdapter::operator[]
(int index) const throw (de-
caf::lang::exceptions::IndexOutOfBoundsException
)`

6.165.3.30 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::put
(int index, unsigned char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes the given byte into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The position in the Buffer to write the data.

value The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.31 `virtual ByteArrayAdapter& de-
caf::internal::util::ByteArrayAdapter::putChar (int index, char
value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
[virtual]`

Writes one byte containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The position in the Buffer to write the data.

value The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.32 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDouble (int index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes eight bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The position in the Buffer to write the data.

value The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.33 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDoubleAt (int index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters

index The position in the Buffer to write the data.

value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.34 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloat (int index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes four bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The position in the Buffer to write the data.

value The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.35 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloatAt (int index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters

index The position in the Buffer to write the data.

value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.36 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putInt (int index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Writes four bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The position in the Buffer to write the data.

value The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.37 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putIntAt (int index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters

index The position in the Buffer to write the data.

value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.38 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLong (int index, long long value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The position in the Buffer to write the data.

value The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.39 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLongAt (int index, long long value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters

index The position in the Buffer to write the data.

value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.40 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShort (int index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Writes two bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index The position in the Buffer to write the data.

value The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.41 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShortAt (int index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Writes two bytes containing the given value, into this buffer at the given byte index.

Parameters

index The position in the Buffer to write the data.

value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.165.3.42 virtual void decaf::internal::util::ByteArrayAdapter::read (unsigned char * *buffer*, int *size*, int *offset*, int *length*) const
throw (decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::NullPointerException,
decaf::nio::BufferUnderflowException) [virtual]

Reads from the Byte array starting at the specified offset and reading the specified length.

If the length is greater than the size of this underlying byte array then an BufferUnderflowException is thrown.

Parameters

buffer The buffer to read data from this array into.

size The size of the buffer passed.

offset The position in this array to start reading from.

length The amount of data to read from this array.

Exceptions

IndexOutOfBoundsException if the offset + length exceeds the size.

NullPointerException if buffer is null

BufferUnderflowException if there is not enough data to read because the offset or the length is greater than the size of this array.

6.165.3.43 virtual void decaf::internal::util::ByteArrayAdapter::resize (int *size*) throw (decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::InvalidStateException) [virtual]

Resizes the underlying array to the new given size, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.

A **ByteArrayAdapter** (p. 893) can only be resized when it owns the underlying array, if it does not then it will throw an InvalidStateException.

Parameters

size The new size of the array.

Exceptions

IllegalArgumentException if the size parameter is negative.

InvalidStateException if this object does not own the buffer.

6.165.3.44 virtual void decaf::internal::util::ByteArrayAdapter::write (unsigned char * *buffer*, int *size*, int *offset*, int *length*)
throw (decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::NullPointerException,
decaf::nio::BufferOverflowException) [virtual]

Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array.

. If the length is greater than the size of this underlying byte array then an `BufferOverflowException` is thrown.

Parameters

- buffer* The buffer to read data from this array into.
- size* The size of the buffer passed.
- offset* The position in this array to start reading from.
- length* The amount of data to read from this array.

Exceptions

- `IndexOutOfBoundsException`* if the `offset + length` exceeds the size.
- `NullPointerException`* if buffer is null
- `BufferOverflowException`* if the amount of data to be written to this array or the offset given are larger than this array's size.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/ByteArrayAdapter.h`

6.166 decaf::internal::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:

```
#include <src/main/decaf/internal/nio/ByteBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::ByteBuffer`:

Public Member Functions

- **ByteBuffer** (int capacity, bool readOnly=false) throw (decaf::lang::exceptions::IllegalArgumentException)
*Creates a **ByteBuffer** (p. 915) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **ByteBuffer** (unsigned char *array, int size, int offset, int length, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a **ByteBuffer** (p. 915) object that wraps the given array.*
- **ByteBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.
- **ByteBuffer** (const ByteBuffer &other)
*Create a **ByteBuffer** (p. 915) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual `~ByteBuffer ()`
- virtual bool `isReadOnly () const`
Tells whether or not this buffer is read-only.
Returns
true if, and only if, this buffer is read-only
- virtual unsigned char * `array () throw (decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException)`
Returns the byte array that backs this buffer.
Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.
Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.
Returns
The array that backs this buffer
Exceptions
***ReadOnlyBufferException** (p. 2966) if this buffer is backed by an array but is read-only*
***UnsupportedOperationException** if this buffer is not backed by an accessible array*
- virtual int `arrayOffset () const throw (decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException)`
Returns the offset within this buffer's backing array of the first element of the buffer.
If this buffer is backed by an array then buffer position `p` corresponds to array index `p + arrayOffset()` (p. 960).
Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.
Returns
The offset within this buffer's array of the first element of the buffer.
Exceptions
***ReadOnlyBufferException** (p. 2966) if this buffer is backed by an array but is read-only.*
***UnsupportedOperationException** if this buffer is not backed by an accessible array.*
- virtual bool `hasArray () const`
Tells whether or not this buffer is backed by an accessible byte array.
If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.
Returns
true if, and only if, this buffer is backed by an array and is not read-only.
- virtual `decaf::nio::CharBuffer * asCharBuffer () const`
Creates a view of this byte buffer as a char buffer.
The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.
The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.
Returns
*the new Char **Buffer** (p. 855), which the caller then owns.*

- virtual **decaf::nio::DoubleBuffer * asDoubleBuffer ()** const

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the new double **Buffer** (p. 855), which the caller then owns.*

- virtual **decaf::nio::FloatBuffer * asFloatBuffer ()** const

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the new float **Buffer** (p. 855), which the caller then owns.*

- virtual **decaf::nio::IntBuffer * asIntBuffer ()** const

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the new int **Buffer** (p. 855), which the caller then owns.*

- virtual **decaf::nio::LongBuffer * asLongBuffer ()** const

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the new long **Buffer** (p. 855), which the caller then owns.*

- virtual **decaf::nio::ShortBuffer * asShortBuffer ()** const

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new short **Buffer** (p. 855), which the caller then owns.

- virtual **ByteBuffer * asReadOnlyBuffer ()** const

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only byte buffer which the caller then owns.

- virtual **ByteBuffer & compact ()** throw (decaf::nio::ReadOnlyBufferException)

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ByteBuffer** (p. 954).

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual **ByteBuffer * duplicate ()**

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new **ByteBuffer** (p. 855) which the caller owns.

- virtual unsigned char **get ()** const throw (decaf::nio::BufferUnderflowException)

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

Returns

The byte at the buffer's current position.

Exceptions

BufferUnderflowException (p. 882) if the buffer's current position is not smaller than its limit.

- virtual unsigned char **get (int index)** const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

Reads the byte at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the byte is to be read.

Returns

the byte that is located at the given index.

Exceptions

IndexOutOfBoundsException if *index* is not smaller than the buffer's limit, or *index* is negative.

- virtual char **getChar** () throw (decaf::nio::BufferUnderflowException)

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns

the next char in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

- virtual char **getChar** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Reads one byte at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the byte is to be read.

Returns

the char at the given index in the buffer

Exceptions

IndexOutOfBoundsException if *index* is not smaller than the buffer's limit, or *index* is negative.

- virtual double **getDouble** () throw (decaf::nio::BufferUnderflowException)

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next double in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

- virtual double **getDouble** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Reads eight bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the double at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if *index* is not smaller than the buffer's limit, or *index* is negative.

- virtual float **getFloat** () throw (decaf::nio::BufferUnderflowException)
Reads the next four bytes at this buffer's current position, and then increments the position by that amount.
Returns
the next float in the buffer.
Exceptions
BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
- virtual float **getFloat** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.
Parameters
index The index in the **Buffer** (p. 855) where the bytes are to be read.
Returns
the float at the given index in the buffer.
Exceptions
IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
- virtual long long **getLong** () throw (decaf::nio::BufferUnderflowException)
Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
Returns
the next long long in the buffer.
Exceptions
BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
- virtual long long **getLong** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given index and returns it.
Parameters
index The index in the **Buffer** (p. 855) where the bytes are to be read.
Returns
the long long at the given index in the buffer.
Exceptions
IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
- virtual int **getInt** () throw (decaf::nio::BufferUnderflowException)
Reads the next four bytes at this buffer's current position, and then increments the position by that amount.
Returns
the next int in the buffer.
Exceptions
BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

- virtual int **getInt** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Reads four bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the int at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

- virtual short **getShort** () throw (decaf::nio::BufferUnderflowException)

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next short in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

- virtual short **getShort** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Reads two bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the short at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

- virtual **ByteBuffer** & **put** (unsigned char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters

value - the byte value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual **ByteBuffer** & **put** (int index, unsigned char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes the given byte into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 855) to write the data
value - the byte to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if *index* greater than the buffer's limit minus the size of the type being written, or *index* is negative.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual **ByteBuffer** & **putChar** (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException (p. 2966) if this buffer is read-only

- virtual **ByteBuffer** & **putChar** (int index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes one byte containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.
value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if *index* greater than the buffer's limit minus the size of the type being written, or *index* is negative.
ReadOnlyBufferException (p. 2966) if this buffer is read-only

- virtual **ByteBuffer** & **putDouble** (double value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException (p. 2966) if this buffer is read-only

- virtual **ByteBuffer** & **putDouble** (int index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data
value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual **ByteBuffer** & **putFloat** (float value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual **ByteBuffer** & **putFloat** (int index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data
value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual **ByteBuffer** & **putLong** (long long value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual **ByteBuffer** & **putLong** (int index, long long value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.
value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual **ByteBuffer** & **putInt** (int value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException (p. 2966) if this buffer is read-only

- virtual **ByteBuffer** & **putInt** (int index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.
value The value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2966) if this buffer is read-only

- virtual **ByteBuffer** & **putShort** (short value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual **ByteBuffer** & **putShort** (int index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes two bytes containing the given value, into this buffer at the given index.

Parameters

index *The position in the **Buffer** (p. 855) to write the data*
value *The value to write.*

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual **ByteBuffer** * **slice** () const

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the newly create **ByteBuffer** (p. 954) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

*Sets this **ByteBuffer** (p. 915) as Read-Only or not Read-Only.*

6.166.1 Detailed Description

This class defines six categories of operations upon byte buffers: 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p.934) float getFloat(int index) void **putFloat(float f)** (p.940) void **putFloat(int index, float f)** (p.940)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The asFloatBuffer method, for example, creates an instance of the FloatBuffer class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

Since

1.0

6.166.2 Constructor & Destructor Documentation

6.166.2.1 decaf::internal::nio::ByteBuffer::ByteBuffer (int *capacity*, bool *readOnly* = false) throw (decaf::lang::exceptions::IllegalArgumentException)

Creates a **ByteBuffer** (p.915) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity The size of the array, this is the limit we read and write to.

readOnly Should this buffer be read-only, default as false

Exceptions

IllegalArgumentException if the capacity value is negative.

6.166.2.2 `decaf::internal::nio::ByteBuffer::ByteBuffer (unsigned char * array, int size, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a **ByteBuffer** (p. 915) object that wraps the given array.

Parameters

array The array to wrap.
size The size of the array passed.
offset The position that is this buffers start position.
length The size of the sub-array, this is the limit we read and write to.
readOnly Should this buffer be read-only, default as false.

Exceptions

NullPointerException if buffer is NULL
IndexOutOfBoundsException if the preconditions of size, offset and length are violated.

6.166.2.3 `decaf::internal::nio::ByteBuffer::ByteBuffer (const decaf::lang::Pointer< ByteBufferAdapter > & array, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed ByteBufferAdapter and start at the given offset.

The capacity and limit of the new **ByteBuffer** (p. 915) will be that of the remaining capacity of the passed buffer.

Parameters

array The ByteBufferAdapter to wrap
offset The offset into array where the buffer starts
length The length of the array we are wrapping or limit.
readOnly Boolean indicating if this a readOnly buffer.

Exceptions

NullPointerException if array is NULL
IndexOutOfBoundsException if offset is greater than array capacity.

6.166.2.4 `decaf::internal::nio::ByteBuffer::ByteBuffer (const ByteBuffer & other)`

Create a **ByteBuffer** (p. 915) that mirrors this one, meaning it shares a reference to this buffers ByteBufferAdapter and when changes are made to that data it is reflected in both.

Parameters

other The **ByteBuffer** (p. 915) this one is to mirror.

6.166.2.5 virtual decaf::internal::nio::ByteBuffer::~~ByteBuffer ()
[virtual]

6.166.3 Member Function Documentation

6.166.3.1 virtual unsigned char* decaf::internal::nio::ByteBuffer::array
() throw (decaf::nio::ReadOnlyBufferException,
decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The array that backs this buffer

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is backed by an array but is read-only
UnsupportedOperationException if this buffer is not backed by an accessible array

Implements decaf::nio::ByteBuffer (p. 960).

6.166.3.2 virtual int decaf::internal::nio::ByteBuffer::arrayOffset
() const throw (decaf::nio::ReadOnlyBufferException,
decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position p corresponds to array index p + **arrayOffset()** (p. 960).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset within this buffer's array of the first element of the buffer.

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is backed by an array but is read-only.
UnsupportedOperationException if this buffer is not backed by an accessible array.

Implements decaf::nio::ByteBuffer (p. 960).

6.166.3.3 virtual decaf::nio::CharBuffer* de-
caaf::internal::nio::ByteBuffer::asCharBuffer ()
const [inline, virtual]

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new Char **Buffer** (p. 855), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 961).

6.166.3.4 `virtual decaf::nio::DoubleBuffer* de-
caf::internal::nio::ByteBuffer::asDoubleBuffer ()
const [inline, virtual]`

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new double **Buffer** (p. 855), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 961).

6.166.3.5 `virtual decaf::nio::FloatBuffer* de-
caf::internal::nio::ByteBuffer::asFloatBuffer ()
const [inline, virtual]`

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new float **Buffer** (p. 855), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 961).

6.166.3.6 `virtual decaf::nio::IntBuffer* decaf::internal::nio::ByteBuffer::asIntBuffer () const [inline, virtual]`

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new int **Buffer** (p. 855), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 962).

6.166.3.7 `virtual decaf::nio::LongBuffer* decaf::internal::nio::ByteBuffer::asLongBuffer () const [inline, virtual]`

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new long **Buffer** (p. 855), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 962).

6.166.3.8 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::asReadOnlyBuffer () const [virtual]`

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only byte buffer which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 962).

6.166.3.9 **virtual decaf::nio::ShortBuffer* decaf::internal::nio::ByteBuffer::asShortBuffer ()**
const [inline, virtual]

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new short **Buffer** (p. 855), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 963).

6.166.3.10 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::compact () throw (decaf::nio::ReadOnlyBufferException)** [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ByteBuffer** (p. 954).

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 963).

6.166.3.11 **virtual ByteBuffer* decaf::internal::nio::ByteBuffer::duplicate ()**
 [virtual]

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new Byte **Buffer** (p. 855) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 963).

6.166.3.12 virtual unsigned char decaf::internal::nio::ByteArrayBuffer::get ()
const throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

Returns

The byte at the buffer's current position.

Exceptions

BufferUnderflowException (p. 882) if the buffer's current position is not smaller than its limit.

Implements **decaf::nio::ByteBuffer** (p. 964).

6.166.3.13 virtual unsigned char decaf::internal::nio::ByteArrayBuffer::get (int *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method.

Reads the byte at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the byte is to be read.

Returns

the byte that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 964).

6.166.3.14 `virtual char decaf::internal::nio::ByteArrayBuffer::getChar () throw (decaf::nio::BufferUnderflowException) [inline, virtual]`

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns

the next char in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 965).

6.166.3.15 `virtual char decaf::internal::nio::ByteArrayBuffer::getChar (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Reads one byte at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the byte is to be read.

Returns

the char at the given index in the buffer

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 966).

6.166.3.16 `virtual double decaf::internal::nio::ByteArrayBuffer::getDouble () throw (decaf::nio::BufferUnderflowException) [virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next double in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 966).

6.166.3.17 virtual double decaf::internal::nio::ByteBuffer::getDouble (int *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads eight bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the double at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 966).

6.166.3.18 virtual float decaf::internal::nio::ByteBuffer::getFloat () throw (decaf::nio::BufferUnderflowException) [virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next float in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 967).

6.166.3.19 virtual float decaf::internal::nio::ByteBuffer::getFloat (int *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads four bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the float at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 967).

6.166.3.20 `virtual int decaf::internal::nio::ByteArrayBuffer::getInt () throw (decaf::nio::BufferUnderflowException) [virtual]`

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next int in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 968).

6.166.3.21 `virtual int decaf::internal::nio::ByteArrayBuffer::getInt (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads four bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the int at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 968).

6.166.3.22 `virtual long long decaf::internal::nio::ByteArrayBuffer::getLong () throw (decaf::nio::BufferUnderflowException) [virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next long long in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 968).

6.166.3.23 `virtual long long decaf::internal::nio::ByteArrayBuffer::getLong (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads eight bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the long long at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 969).

6.166.3.24 `virtual short decaf::internal::nio::ByteArrayBuffer::getShort (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads two bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the short at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 969).

6.166.3.25 `virtual short decaf::internal::nio::ByteArrayBuffer::getShort () throw (decaf::nio::BufferUnderflowException) [virtual]`

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next short in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 969).

6.166.3.26 `virtual bool decaf::internal::nio::ByteArrayBuffer::hasArray () const`
`[inline, virtual]`

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements `decaf::nio::ByteBuffer` (p. 970).

6.166.3.27 `virtual bool decaf::internal::nio::ByteArrayBuffer::isReadOnly ()`
`const [inline, virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

Implements `decaf::nio::ByteBuffer` (p. 970).

6.166.3.28 `virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::put (`
`unsigned char value) throw (decaf::nio::BufferOverflowException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters

value - the byte value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements `decaf::nio::ByteBuffer` (p. 970).

6.166.3.29 `virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::put`
`(int index, unsigned char value) throw (`
`decaf::lang::exceptions::IndexOutOfBoundsException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given byte into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 855) to write the data
value - the byte to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 971).

6.166.3.30 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putChar (int index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes one byte containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.
value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 973).

6.166.3.31 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putChar (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 973).

6.166.3.32 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putDouble (int index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)` [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data

value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 974).

6.166.3.33 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putDouble (double value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)` [virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 973).

6.166.3.34 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat (int *index*, float *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data
value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 975).

6.166.3.35 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat (float *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 974).

6.166.3.36 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt (int *index*, int *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 975).

6.166.3.37 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt (int value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 975).

6.166.3.38 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putLong (long long value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 976).

6.166.3.39 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putLong (int *index*, long long *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 976).

6.166.3.40 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putShort (short *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 977).

6.166.3.41 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putShort (int index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes two bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data
value The value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 977).

6.166.3.42 `virtual void decaf::internal::nio::ByteBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **ByteBuffer** (p. 915) as Read-Only or not Read-Only.

Parameters

value Boolean value, true if this buffer is to be read-only, false otherwise.

6.166.3.43 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::slice () const [virtual]`

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ByteBuffer** (p. 954) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 977).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ByteBuffer.h`

6.167 decaf::io::ByteArrayInputStream Class Reference

A **ByteArrayInputStream** (p. 944) contains an internal buffer that contains bytes that may be read from the stream.

```
#include <src/main/decaf/io/ByteArrayInputStream.h>
```

Inheritance diagram for decaf::io::ByteArrayInputStream:

Public Member Functions

- **ByteArrayInputStream** ()
*Creates an **ByteArrayInputStream** (p. 944) with an empty input buffer, the buffer can be initialized with a call to **setByteArray**.*
- **ByteArrayInputStream** (const std::vector< unsigned char > &buffer)
*Creates the input stream and calls **setBuffer** with the specified buffer object.*
- **ByteArrayInputStream** (const unsigned char *buffer, int bufferSize, bool own=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
*Create an instance of the **ByteArrayInputStream** (p. 944) with the given buffer as the source of input for all read operations.*
- **ByteArrayInputStream** (const unsigned char *buffer, int bufferSize, int offset, int length, bool own=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
*Create an instance of the **ByteArrayInputStream** (p. 944) with the given buffer as the source of input for all read operations.*
- virtual ~**ByteArrayInputStream** ()
- virtual void **setByteArray** (const std::vector< unsigned char > &buffer)
Sets the internal buffer.
- virtual void **setByteArray** (const unsigned char *buffer, int bufferSize) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.
- virtual void **setByteArray** (const unsigned char *buffer, int bufferSize, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.
- virtual int **available** () const throw (IOException)
*Indicates the number of bytes available.
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.
The default implementation of this method returns zero.*

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

- virtual long long **skip** (long long num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

*The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

Parameters

num *The number of bytes to skip.*

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

- virtual void **mark** (int readLimit)

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit *The max bytes read before marked position is invalid.*

- virtual void **reset** () throw (IOException)

Repositions this stream to the position at the time the mark method was last called on this input stream.

*If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2003) might be thrown. * If such an **IOException** (p. 2003) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.*

*If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 2003). * If an **IOException** (p. 2003) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 2003).*

Exceptions

IOException (p. 2003) if an I/O error occurs.

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Protected Member Functions

- virtual int **doReadByte** () throw (IOException)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsExpection, decaf::lang::exceptions::NullPointerException)

6.167.1 Detailed Description

A **ByteArrayInputStream** (p. 944) contains an internal buffer that contains bytes that may be read from the stream. An internal counter keeps track of the next byte to be supplied by the read method. The **ByteArrayInputStream** (p. 944) never copies the supplied buffers, only points to them, therefore the caller must ensure that the supplied buffer remain in scope, or is not deleted before this **ByteArrayInputStream** (p. 944) is freed. If the own argument of one of the constructors that accepts an array pointer is set to true than the **ByteArrayInputStream** (p. 944) instance will take ownership of the supplied pointer and delete it when that instance is destroyed.

Closing a **ByteArrayInputStream** (p. 944) has no effect. The methods in this class can be called after the stream has been closed without generating an **IOException** (p. 2003).

Since

1.0

6.167.2 Constructor & Destructor Documentation

6.167.2.1 decaf::io::ByteArrayInputStream::ByteArrayInputStream ()

Creates an **ByteArrayInputStream** (p. 944) with an empty input buffer, the buffer can be initialized with a call to **setByteArray**.

6.167.2.2 decaf::io::ByteArrayInputStream::ByteArrayInputStream (const std::vector< unsigned char > & *buffer*)

Creates the input stream and calls **setBuffer** with the specified buffer object.

Parameters

buffer The buffer to be used.

6.167.2.3 `decaf::io::ByteArrayInputStream::ByteArrayInputStream (const unsigned char * buffer, int bufferSize, bool own = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)`

Create an instance of the **ByteArrayInputStream** (p. 944) with the given buffer as the source of input for all read operations.

Parameters

buffer The initial byte array to use to read from.

bufferSize The size of the buffer.

own Indicates if this object should take ownership of the array, default is false.

Exceptions

NullPointerException if the buffer is Null.

IllegalArgumentException if the bufferSize is negative.

6.167.2.4 `decaf::io::ByteArrayInputStream::ByteArrayInputStream (const unsigned char * buffer, int bufferSize, int offset, int length, bool own = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)`

Create an instance of the **ByteArrayInputStream** (p. 944) with the given buffer as the source of input for all read operations.

Parameters

buffer The initial byte array to use to read from.

bufferSize The size of the buffer.

offset The offset into the buffer to begin reading from.

length The number of bytes to read past the offset.

own Indicates if this object should take ownership of the array, default is false.

Exceptions

NullPointerException if the buffer is Null.

IllegalArgumentException if the bufferSize is negative.

6.167.2.5 `virtual decaf::io::ByteArrayInputStream::~~ByteArrayInputStream () [virtual]`

6.167.3 Member Function Documentation

6.167.3.1 `virtual int decaf::io::ByteArrayInputStream::available () const throw (IOException) [virtual]`

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1911).

6.167.3.2 `virtual int decaf::io::ByteArrayInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)` [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1912).

6.167.3.3 `virtual int decaf::io::ByteArrayInputStream::doReadByte () throw (IOException)` [protected, virtual]

Implements **decaf::io::InputStream** (p. 1912).

6.167.3.4 `virtual void decaf::io::ByteArrayInputStream::mark (int readLimit)` [virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit The max bytes read before marked position is invalid.

Reimplemented from **decaf::io::InputStream** (p. 1913).

6.167.3.5 `virtual bool decaf::io::ByteArrayInputStream::markSupported () const` [inline, virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::InputStream** (p. 1913).

6.167.3.6 virtual void decaf::io::ByteArrayInputStream::reset () throw (IOException) [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2003) might be thrown. * If such an **IOException** (p. 2003) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 2003). * If an **IOException** (p. 2003) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2003).

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1916).

6.167.3.7 virtual void decaf::io::ByteArrayInputStream::setByteArray (const unsigned char * buffer, int bufferSize) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException) [virtual]

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

Parameters

buffer The initial byte array to use to read from.

bufferSize The size of the buffer.

Exceptions

NullPointerException if the buffer is Null.

IllegalArgumentException if the bufferSize is negative.

6.167.3.8 virtual void decaf::io::ByteArrayInputStream::setByteArray (const std::vector< unsigned char > & *buffer*) [virtual]

Sets the internal buffer.

The input stream will wrap around this buffer and will perform all read operations on it. The position will be reinitialized to the beginning of the specified buffer. This class will not own the given buffer - it is the caller's responsibility to free the memory of the given buffer as appropriate.

Parameters

buffer The buffer to be used.

6.167.3.9 virtual void decaf::io::ByteArrayInputStream::setByteArray (const unsigned char * *buffer*, int *bufferSize*, int *offset*, int *length*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException) [virtual]

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

Parameters

buffer The initial byte array to use to read from.

bufferSize The size of the buffer.

offset The offset into the buffer to begin reading from.

length The number of bytes to read past the offset.

Exceptions

NullPointerException if the buffer is Null.

IllegalArgumentException if the bufferSize is negative.

6.167.3.10 virtual long long decaf::io::ByteArrayInputStream::skip (long long *num*) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1917).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/ByteArrayInputStream.h`

6.168 decaf::io::ByteArrayOutputStream Class Reference

`#include <src/main/decaf/io/ByteArrayOutputStream.h>`

Inheritance diagram for `decaf::io::ByteArrayOutputStream`:

Public Member Functions

- **ByteArrayOutputStream** ()
Default Constructor - uses a default internal buffer of 32 bytes, the size increases as the need for more room arises.
- **ByteArrayOutputStream** (int bufferSize) throw (decaf::lang::exceptions::IllegalArgumentException)
*Creates a **ByteArrayOutputStream** (p. 951) with an internal buffer allocated with the given size.*
- virtual **~ByteArrayOutputStream** ()
- `std::pair< unsigned char *, int >` **toByteArray** () const
Creates a newly allocated byte array.
- long long **size** () const
*Gets the current count of bytes written into this **ByteArrayOutputStream** (p. 951).*
- virtual void **reset** () throw (IOException)
Clear current Stream contents.
- virtual `std::string` **toString** () const
Converts the bytes in the buffer into a standard C++ string.
- void **writeTo** (OutputStream *out) const throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using `out.write(buf, 0, count)`.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw (decaf::io::IOException)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

6.168.1 Constructor & Destructor Documentation

6.168.1.1 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream ()

Default Constructor - uses a default internal buffer of 32 bytes, the size increases as the need for more room arises.

6.168.1.2 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream (int *bufferSize*) throw (decaf::lang::exceptions::IllegalArgumentException)

Creates a **ByteArrayOutputStream** (p. 951) with an internal buffer allocated with the given size.

Parameters

bufferSize The size to use for the internal buffer.

Exceptions

IllegalArgumentException if the size is less than or equal to zero.

6.168.1.3 virtual decaf::io::ByteArrayOutputStream::~~ByteArrayOutputStream () [virtual]

6.168.2 Member Function Documentation

6.168.2.1 virtual void decaf::io::ByteArrayOutputStream::doWriteArrayBounded (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2721).

6.168.2.2 virtual void decaf::io::ByteArrayOutputStream::doWriteByte (unsigned char *value*) throw (decaf::io::IOException) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2721).

6.168.2.3 virtual void decaf::io::ByteArrayOutputStream::reset () throw (IOException) [virtual]

Clear current Stream contents.

Exceptions

IOException (p. 2003)

6.168.2.4 long long decaf::io::ByteArrayOutputStream::size () const

Gets the current count of bytes written into this **ByteArrayOutputStream** (p. 951).

Returns

the number of valid bytes contained in the **ByteArrayOutputStream** (p. 951).

6.168.2.5 std::pair<unsigned char*, int> decaf::io::ByteArrayOutputStream::toByteArray () const

Creates a newly allocated byte array.

Its size is the current size of this output stream and the valid contents of the buffer have been copied into it. The newly allocated array and its size are returned inside an STL pair structure, the caller is responsible for freeing the returned array.

Returns

an STL pair containing the copied array and its size.

6.168.2.6 virtual std::string decaf::io::ByteArrayOutputStream::toString () const [virtual]

Converts the bytes in the buffer into a standard C++ string.

Returns

a string containing the bytes in the buffer

Reimplemented from **decaf::io::OutputStream** (p. 2722).

6.168.2.7 void decaf::io::ByteArrayOutputStream::writeTo (OutputStream * out) const throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using `out.write(buf, 0, count)`.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/ByteArrayOutputStream.h`

6.169 decaf::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:

```
#include <src/main/decaf/nio/ByteBuffer.h>
```

Inheritance diagram for decaf::nio::ByteBuffer:

Public Member Functions

- virtual **~ByteBuffer** ()
- virtual std::string **toString** () const
- **ByteBuffer** & **get** (std::vector< unsigned char > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **ByteBuffer** & **get** (unsigned char *buffer, int size, int offset, int length) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)
Relative bulk get method.
- **ByteBuffer** & **put** (**ByteBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException)
This method transfers the bytes remaining in the given source buffer into this buffer.
- **ByteBuffer** & **put** (const unsigned char *buffer, int size, int offset, int length) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)
This method transfers bytes into this buffer from the given source array.
- **ByteBuffer** & **put** (std::vector< unsigned char > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source byte array into this buffer.
- virtual bool **isReadOnly** () const =0
Tells whether or not this buffer is read-only.
- virtual unsigned char * **array** ()=0 throw (ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException)
Returns the byte array that backs this buffer.
- virtual int **arrayOffset** () const =0 throw (ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException)
Returns the offset within this buffer's backing array of the first element of the buffer.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible byte array.
- virtual **CharBuffer** * **asCharBuffer** () const =0

Creates a view of this byte buffer as a char buffer.

- virtual **DoubleBuffer** * **asDoubleBuffer** () const =0
Creates a view of this byte buffer as a double buffer.
- virtual **FloatBuffer** * **asFloatBuffer** () const =0
Creates a view of this byte buffer as a float buffer.
- virtual **IntBuffer** * **asIntBuffer** () const =0
Creates a view of this byte buffer as a int buffer.
- virtual **LongBuffer** * **asLongBuffer** () const =0
Creates a view of this byte buffer as a long buffer.
- virtual **ShortBuffer** * **asShortBuffer** () const =0
Creates a view of this byte buffer as a short buffer.
- virtual **ByteBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only byte buffer that shares this buffer's content.
- virtual **ByteBuffer** & **compact** ()=0 throw (**ReadOnlyBufferException**)
Compacts this buffer.
- virtual **ByteBuffer** * **duplicate** ()=0
Creates a new byte buffer that shares this buffer's content.
- virtual unsigned char **get** () const =0 throw (**BufferUnderflowException**)
Relative get method.
- virtual unsigned char **get** (int index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- virtual char **getChar** ()=0 throw (**BufferUnderflowException**)
Reads the next byte at this buffer's current position, and then increments the position by one.
- virtual char **getChar** (int index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads one byte at the given index and returns it.
- virtual double **getDouble** ()=0 throw (**BufferUnderflowException**)
Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
- virtual double **getDouble** (int index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given index and returns it.
- virtual float **getFloat** ()=0 throw (**BufferUnderflowException**)
Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

- virtual float **getFloat** (int index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.
- virtual long long **getLong** ()=0 throw (BufferUnderflowException)
Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
- virtual long long **getLong** (int index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given index and returns it.
- virtual int **getInt** ()=0 throw (BufferUnderflowException)
Reads the next four bytes at this buffer's current position, and then increments the position by that amount.
- virtual int **getInt** (int index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.
- virtual short **getShort** ()=0 throw (BufferUnderflowException)
Reads the next two bytes at this buffer's current position, and then increments the position by that amount.
- virtual short **getShort** (int index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads two bytes at the given index and returns it.
- virtual **ByteBuffer** & **put** (unsigned char value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given byte into this buffer at the current position, and then increments the position.
- virtual **ByteBuffer** & **put** (int index, unsigned char value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given byte into this buffer at the given index.
- virtual **ByteBuffer** & **putChar** (char value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.
- virtual **ByteBuffer** & **putChar** (int index, char value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes one byte containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putDouble** (double value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putDouble** (int index, double value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putFloat** (float value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putFloat** (int index, float value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes four bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putLong** (long long value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putLong** (int index, long long value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putInt** (int value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putInt** (int index, int value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes four bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putShort** (short value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putShort** (int index, short value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes two bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** * **slice** () const =0
Creates a new byte buffer whose content is a shared subsequence of this buffer's content.
- virtual int **compareTo** (const **ByteBuffer** &value) const
- virtual bool **equals** (const **ByteBuffer** &value) const
- virtual bool **operator==** (const **ByteBuffer** &value) const

- virtual bool **operator**< (const **ByteBuffer** &value) const

Static Public Member Functions

- static **ByteBuffer** * **allocate** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)
Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **ByteBuffer** * **wrap** (unsigned char *array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
*Wraps the passed buffer with a new **ByteBuffer** (p. 954).*
- static **ByteBuffer** * **wrap** (std::vector< unsigned char > &buffer)
*Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 954).*

Protected Member Functions

- **ByteBuffer** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)
*Creates a **ByteBuffer** (p. 954) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.169.1 Detailed Description

This class defines six categories of operations upon byte buffers: 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p.967) float **getFloat**(int index) void **putFloat(float f)** (p.974) void **putFloat**(int index, float f) (p.975)

Corresponding methods are defined for the types `char`, `short`, `int`, `long`, and `double`. The index parameters of the absolute `get` and `put` methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The `asFloatBuffer` method, for example, creates an instance of the **FloatBuffer** (p.1800) class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types `char`, `short`, `int`, `long`, and `double`.

View buffers have two important advantages over the families of type-specific `get` and `put` methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk `get` and `put` methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

6.169.2 Constructor & Destructor Documentation

6.169.2.1 `decaf::nio::ByteBuffer::ByteBuffer (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException) [protected]`

Creates a **ByteBuffer** (p.954) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity The size of the array, this is the limit we read and write to.

Exceptions

IllegalArgumentException if *capacity* is negative.

6.169.2.2 `virtual decaf::nio::ByteBuffer::~~ByteBuffer () [inline, virtual]`

6.169.3 Member Function Documentation

6.169.3.1 `static ByteBuffer* decaf::nio::ByteBuffer::allocate (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException) [static]`

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity The internal buffer's capacity.

Returns

a newly allocated **ByteBuffer** (p.954) which the caller owns.

Exceptions

IllegalArgumentException if capacity is negative.

6.169.3.2 `virtual unsigned char* decaf::nio::ByteBuffer::array
() throw (ReadOnlyBufferException, de-
caf::lang::exceptions::UnsupportedOperationException) [pure
virtual]`

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The array that backs this buffer

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is backed by an array but is read-only

UnsupportedOperationException if this buffer is not backed by an accessible array

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 928).

6.169.3.3 `virtual int decaf::nio::ByteBuffer::arrayOffset
() const throw (ReadOnlyBufferException,
decaf::lang::exceptions::UnsupportedOperationException) [pure
virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position `p` corresponds to array index `p + arrayOffset()` (p. 960).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset within this buffer's array of the first element of the buffer.

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is backed by an array but is read-only.

UnsupportedOperationException if this buffer is not backed by an accessible array.

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 928).

6.169.3.4 `virtual CharBuffer* decaf::nio::ByteBuffer::asCharBuffer () const` [pure virtual]

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new Char **Buffer** (p. 855), which the caller then owns.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 928).

6.169.3.5 `virtual DoubleBuffer* decaf::nio::ByteBuffer::asDoubleBuffer () const` [pure virtual]

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new double **Buffer** (p. 855), which the caller then owns.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 929).

6.169.3.6 `virtual FloatBuffer* decaf::nio::ByteBuffer::asFloatBuffer () const` [pure virtual]

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new float **Buffer** (p. 855), which the caller then owns.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 929).

6.169.3.7 virtual IntBuffer* decaf::nio::ByteBuffer::asIntBuffer () const [pure virtual]

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new int **Buffer** (p. 855), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 930).

6.169.3.8 virtual LongBuffer* decaf::nio::ByteBuffer::asLongBuffer () const [pure virtual]

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new long **Buffer** (p. 855), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 930).

6.169.3.9 virtual ByteBuffer* decaf::nio::ByteBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only byte buffer which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 930).

6.169.3.10 `virtual ShortBuffer* decaf::nio::ByteBuffer::asShortBuffer () const`
`[pure virtual]`

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new short **Buffer** (p. 855), which the caller then owns.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 931).

6.169.3.11 `virtual ByteBuffer& decaf::nio::ByteBuffer::compact () throw (`
`ReadOnlyBufferException) [pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 860) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 859) - 1 is copied to index `n = limit()` (p. 859) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ByteBuffer** (p. 954).

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 931).

6.169.3.12 `virtual int decaf::nio::ByteBuffer::compareTo (const ByteBuffer &`
`value) const [virtual]`

6.169.3.13 `virtual ByteBuffer* decaf::nio::ByteBuffer::duplicate () [pure`
`virtual]`

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new Byte **Buffer** (p. 855) which the caller owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 931).

6.169.3.14 `virtual bool decaf::nio::ByteBuffer::equals (const ByteBuffer & value) const` [virtual]

6.169.3.15 `ByteBuffer& decaf::nio::ByteBuffer::get (std::vector< unsigned char > buffer) throw (BufferUnderflowException)`

Relative bulk get method.

This method transfers bytes from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this Byte **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than length bytes remaining in this buffer

6.169.3.16 `virtual unsigned char decaf::nio::ByteBuffer::get () const throw (BufferUnderflowException)` [pure virtual]

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

Returns

The byte at the buffer's current position.

Exceptions

BufferUnderflowException (p. 882) if the buffer's current position is not smaller than its limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 932).

6.169.3.17 `virtual unsigned char decaf::nio::ByteBuffer::get (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Absolute get method.

Reads the byte at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the byte is to be read.

Returns

the byte that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 932).

6.169.3.18 **ByteBuffer& decaf::nio::ByteBuffer::get (unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)**

Relative bulk get method.

This method transfers bytes from this buffer into the given destination array. If there are fewer bytes remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 860), then no bytes are transferred and a **BufferUnderflowException** (p. 882) is thrown.

Otherwise, this method copies `length` bytes from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters

buffer The pointer to an allocated buffer to fill.

size The size of the passed in **Buffer** (p. 855).

offset The position in the buffer to start filling.

length The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

IndexOutOfBoundsException if the preconditions of size, offset, or length are not met.

BufferUnderflowException (p. 882) if there are fewer than `length` bytes remaining in this buffer.

NullPointerException if the passed buffer is null.

6.169.3.19 **virtual char decaf::nio::ByteBuffer::getChar () throw (BufferUnderflowException) [pure virtual]**

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns

the next char in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 933).

6.169.3.20 `virtual char decaf::nio::ByteBuffer::getChar (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Reads one byte at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the byte is to be read.

Returns

the char at the given index in the buffer

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 933).

6.169.3.21 `virtual double decaf::nio::ByteBuffer::getDouble () throw (BufferUnderflowException) [pure virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next double in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 933).

6.169.3.22 `virtual double decaf::nio::ByteBuffer::getDouble (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Reads eight bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the double at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 934).

6.169.3.23 `virtual float decaf::nio::ByteBuffer::getFloat () throw (BufferUnderflowException) [pure virtual]`

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next float in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 934).

6.169.3.24 `virtual float decaf::nio::ByteBuffer::getFloat (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Reads four bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the float at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 934).

6.169.3.25 virtual int decaf::nio::ByteBuffer::getInt () throw (
BufferUnderflowException) [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next int in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 935).

6.169.3.26 virtual int decaf::nio::ByteBuffer::getInt (int *index*) const throw (
decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Reads four bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the int at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 935).

6.169.3.27 virtual long long decaf::nio::ByteBuffer::getLong () throw (
BufferUnderflowException) [pure virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next long long in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 935).

6.169.3.28 `virtual long long decaf::nio::ByteBuffer::getLong (int index) const
throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure
virtual]`

Reads eight bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the long long at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 936).

6.169.3.29 `virtual short decaf::nio::ByteBuffer::getShort () throw (
BufferUnderflowException) [pure virtual]`

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next short in the buffer.

Exceptions

BufferUnderflowException (p. 882) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 936).

6.169.3.30 `virtual short decaf::nio::ByteBuffer::getShort (int index) const
throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure
virtual]`

Reads two bytes at the given index and returns it.

Parameters

index The index in the **Buffer** (p. 855) where the bytes are to be read.

Returns

the short at the given index in the buffer.

Exceptions

IndexOutOfBoundsException if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 936).

6.169.3.31 `virtual bool decaf::nio::ByteBuffer::hasArray () const [pure virtual]`

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 937).

6.169.3.32 `virtual bool decaf::nio::ByteBuffer::isReadOnly () const [pure virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 937).

6.169.3.33 `virtual bool decaf::nio::ByteBuffer::operator< (const ByteBuffer & value) const [virtual]`**6.169.3.34** `virtual bool decaf::nio::ByteBuffer::operator== (const ByteBuffer & value) const [virtual]`**6.169.3.35** `virtual ByteBuffer& decaf::nio::ByteBuffer::put (unsigned char value) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]`

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters

value - the byte value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 937).

6.169.3.36 `virtual ByteBuffer& decaf::nio::ByteBuffer::put
(int index, unsigned char value) throw (
decaf::lang::exceptions::IndexOutOfBoundsException,
ReadOnlyBufferException) [pure virtual]`

Writes the given byte into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 855) to write the data

value - the byte to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 937).

6.169.3.37 `ByteBuffer& decaf::nio::ByteBuffer::put (ByteBuffer & src)
throw (BufferOverflowException, ReadOnlyBufferException,
decaf::lang::exceptions::IllegalArgumentException)`

This method transfers the bytes remaining in the given source buffer into this buffer.

If there are more bytes remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 860), then no bytes are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies `n = src.remaining()` bytes from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

src The buffer to take bytes from an place in this one.

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer for the remaining bytes in the source buffer

IllegalArgumentException if the source buffer is this buffer

ReadOnlyBufferException (p. 2966) if this buffer is read-only

6.169.3.38 `ByteBuffer& decaf::nio::ByteBuffer::put (const unsigned char * buffer, int size, int offset, int length) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)`

This method transfers bytes into this buffer from the given source array.

If there are more bytes to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 860), then no bytes are transferred and a `BufferOverflowException` (p. 880) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

buffer The array from which bytes are to be read.

size The size of the given array.

offset The offset within the array of the first byte to be read.

length The number of bytes to be read from the given array.

Returns

a reference to this buffer.

Exceptions

`BufferOverflowException` (p. 880) if there is insufficient space in this buffer.

`ReadOnlyBufferException` (p. 2966) if this buffer is read-only.

`NullPointerException` if the passed buffer is null.

`IndexOutOfBoundsException` if the preconditions of `size`, `offset`, or `length` are not met.

6.169.3.39 `ByteBuffer& decaf::nio::ByteBuffer::put (std::vector< unsigned char > & buffer) throw (BufferOverflowException, ReadOnlyBufferException)`

This method transfers the entire content of the given source byte array into this buffer.

This is the same as calling `put(&buffer[0], buffer.size(), 0, buffer.size())`

Parameters

buffer The buffer whose contents are copied to this `ByteBuffer` (p. 954).

Returns

a reference to this buffer.

Exceptions

`BufferOverflowException` (p. 880) if there is insufficient space in this buffer.

`ReadOnlyBufferException` (p. 2966) if this buffer is read-only.

6.169.3.40 `virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (char value)
throw (BufferOverflowException, ReadOnlyBufferException) [pure
virtual]`

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 938).

6.169.3.41 `virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (int index, char
value) throw (decaf::lang::exceptions::IndexOutOfBoundsException,
ReadOnlyBufferException) [pure virtual]`

Writes one byte containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 938).

6.169.3.42 `virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (double value
) throw (BufferOverflowException, ReadOnlyBufferException) [pure
virtual]`

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 939).

6.169.3.43 virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble
(int *index*, double *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException,
ReadOnlyBufferException) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data

value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 939).

6.169.3.44 virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (float *value*)
throw (BufferOverflowException, ReadOnlyBufferException) [pure
virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 940).

6.169.3.45 `virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (int index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data

value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 940).

6.169.3.46 `virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (int index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 940).

6.169.3.47 `virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (int value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 941).

6.169.3.48 virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (long long *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 941).

6.169.3.49 virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (int *index*, long long *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 942).

6.169.3.50 `virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (int index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]`

Writes two bytes containing the given value, into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data
value The value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 943).

6.169.3.51 `virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (short value) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]`

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value The value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 942).

6.169.3.52 `virtual ByteBuffer* decaf::nio::ByteBuffer::slice () const [pure virtual]`

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ByteBuffer** (p. 954) which the caller owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 943).

6.169.3.53 virtual std::string decaf::nio::ByteBuffer::toString () const [virtual]

Returns

a std::string describing this object

6.169.3.54 static ByteBuffer* decaf::nio::ByteBuffer::wrap (unsigned char * *array*, int *size*, int *offset*, int *length*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [static]

Wraps the passed buffer with a new **ByteBuffer** (p. 954).

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array The array that will back the new buffer.

size The size of the provided array.

offset The offset of the subarray to be used.

length The length of the subarray to be used.

Returns

a new **ByteBuffer** (p. 954) that is backed by buffer, caller owns.

Exceptions

NullPointerException if the array passed in is NULL.

IndexOutOfBoundsException if the preconditions of size, offset, or length are not met.

6.169.3.55 static ByteBuffer* decaf::nio::ByteBuffer::wrap (std::vector< unsigned char > & *buffer*) [static]

Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 954).

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new **ByteBuffer** (p. 954) that is backed by *buffer*, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/ByteBuffer.h`

6.170 cms::BytesMessage Class Reference

A **BytesMessage** (p. 979) object is used to send a message containing a stream of unsigned bytes.

```
#include <src/main/cms/BytesMessage.h>
```

Inheritance diagram for `cms::BytesMessage`:

Public Member Functions

- virtual `~BytesMessage ()`
- virtual void **setBodyBytes** (const unsigned char *buffer, int numBytes)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
sets the bytes given to the message body.
- virtual unsigned char * **getBodyBytes** () const =0 throw (cms::MessageNotReadableException, cms::CMSEException)
Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.
- virtual int **getBodyLength** () const =0 throw (cms::MessageNotReadableException, cms::CMSEException)
Returns the number of bytes contained in the body of this message.
- virtual void **reset** ()=0 throw (cms::MessageFormatException, cms::CMSEException)
Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Boolean from the Bytes message stream.
- virtual void **writeBoolean** (bool value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a boolean to the bytes message stream as a 1-byte value.
- virtual unsigned char **readByte** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a Byte from the Bytes message stream.

- virtual void **writeByte** (unsigned char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a byte to the bytes message stream as a 1-byte value.

- virtual int **readBytes** (std::vector< unsigned char > &value) const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a byte array from the bytes message stream.

- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

- virtual int **readBytes** (unsigned char *buffer, int length) const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a portion of the bytes message stream.

- virtual void **writeBytes** (const unsigned char *value, int offset, int length)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a portion of a byte array to the bytes message stream.

- virtual char **readChar** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a Char from the Bytes message stream.

- virtual void **writeChar** (char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a char to the bytes message stream as a 1-byte value.

- virtual float **readFloat** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit float from the Bytes message stream.

- virtual void **writeFloat** (float value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a float to the bytes message stream as a 4 byte value.

- virtual double **readDouble** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit double from the Bytes message stream.

- virtual void **writeDouble** (double value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a double to the bytes message stream as a 8 byte value.

- virtual short **readShort** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit signed short from the Bytes message stream.

- virtual void **writeShort** (short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed short to the bytes message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit unsigned short from the Bytes message stream.

- virtual void **writeUnsignedShort** (unsigned short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a unsigned short to the bytes message stream as a 2 byte value.

- virtual int **readInt** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit signed integer from the Bytes message stream.

- virtual void **writeInt** (int value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed int to the bytes message stream as a 4 byte value.

- virtual long long **readLong** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit long from the Bytes message stream.

- virtual void **writeLong** (long long value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a long long to the bytes message stream as a 8 byte value.

- virtual std::string **readString** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads an ASCII String from the Bytes message stream.

- virtual void **writeString** (const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes an ASCII String to the Bytes message stream.

- virtual std::string **readUTF** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

*Reads an UTF String from the **BytesMessage** (p. 979) stream.*

- virtual void **writeUTF** (const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

*Writes an UTF String to the **BytesMessage** (p. 979) stream.*

- virtual **BytesMessage** * **clone** () const =0

Clones this message.

6.170.1 Detailed Description

A **BytesMessage** (p. 979) object is used to send a message containing a stream of unsigned bytes. It inherits from the **Message** (p. 2375) interface and adds a bytes message body. The receiver of the message supplies the interpretation of the bytes using the methods added by the **BytesMessage** (p. 979) interface.

The **BytesMessage** (p. 979) methods are based largely on those found in **decaf.io.DataInputStream** (p. 1460) and **decaf.io.DataOutputStream** (p. 1473).

Although the CMS API allows the use of message properties with byte messages, they are typically not used, since the inclusion of properties may affect the format.

The primitive types can be written explicitly using methods for each type. Because the C++ language is more limited when dealing with primitive types the JMS equivalent generic read and write methods that take Java objects cannot be provided in the CMS API.

When the message is first created, and when `clearBody` is called, the body of the message is in write-only mode. After the first call to `reset` has been made, the message body is in read-only mode. After a message has been sent, the client that sent it can retain and modify it without affecting the message that has been sent. The same message object can be sent multiple times. When a message has been received, the provider has called `reset` so that the message body is in read-only mode for the client.

If `clearBody` is called on a message in read-only mode, the message body is cleared and the message is in write-only mode.

If a client attempts to read a message in write-only mode, a **MessageNotReadableException** (p. 2548) is thrown.

If a client attempts to write a message in read-only mode, a **MessageNotWriteableException** (p. 2549) is thrown.

Since

1.0

6.170.2 Constructor & Destructor Documentation

6.170.2.1 `virtual cms::BytesMessage::~BytesMessage () [inline, virtual]`

6.170.3 Member Function Documentation

6.170.3.1 `virtual BytesMessage* cms::BytesMessage::clone () const [pure virtual]`

Clones this message.

Returns

a deep copy of this message.

Exceptions

CMSException (p. 1074) - if an internal error occurs while cloning the **Message** (p. 2375).

Implements **cms::Message** (p. 2380).

6.170.3.2 `virtual unsigned char* cms::BytesMessage::getBodyBytes () const
throw (cms::MessageNotReadableException, cms::CMSException)
[pure virtual]`

Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.

This is a copy of the data contained in this message, changing the value contained in this array has no effect on the data contained in this message.

Returns

pointer to a byte buffer that the call owns upon completion of this method.

Exceptions

CMSException (p. 1074) - If an internal error occurs.

MessageNotReadableException (p. 2548) - If the message is in Write Only Mode.

6.170.3.3 `virtual int cms::BytesMessage::getBodyLength () const throw (cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Returns the number of bytes contained in the body of this message.

Returns

number of bytes.

Exceptions

CMSException (p. 1074) - If an internal error occurs.

MessageNotReadableException (p. 2548) - If the message is in Write Only Mode.

6.170.3.4 `virtual bool cms::BytesMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a Boolean from the Bytes message stream.

Returns

boolean value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.5 virtual unsigned char cms::BytesMessage::readByte () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a Byte from the Bytes message stream.

Returns

unsigned char value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.6 virtual int cms::BytesMessage::readBytes (std::vector< unsigned char > & value) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a byte array from the bytes message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

value buffer to place data in

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.7 virtual int cms::BytesMessage::readBytes (unsigned char * buffer, int length) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a portion of the bytes message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

Parameters

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.8 `virtual char cms::BytesMessage::readChar () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a Char from the Bytes message stream.

Returns

char value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.9 `virtual double cms::BytesMessage::readDouble () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 64 bit double from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.10 `virtual float cms::BytesMessage::readFloat () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException)` [pure virtual]

Reads a 32 bit float from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.11 `virtual int cms::BytesMessage::readInt () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException)` [pure virtual]

Reads a 32 bit signed integer from the Bytes message stream.

Returns

int value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.12 `virtual long long cms::BytesMessage::readLong () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException)` [pure virtual]

Reads a 64 bit long from the Bytes message stream.

Returns

long long value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.13 virtual short cms::BytesMessage::readShort () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a 16 bit signed short from the Bytes message stream.

Returns

short value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.14 virtual std::string cms::BytesMessage::readString () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads an ASCII String from the Bytes message stream.

Returns

String from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.15 virtual unsigned short cms::BytesMessage::readUnsignedShort () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

Returns

unsigned short value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.16 virtual std::string cms::BytesMessage::readUTF () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads an UTF String from the BytesMessage (p. 979) stream.

Returns

String from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.170.3.17 virtual void cms::BytesMessage::reset () throw (cms::MessageFormatException, cms::CMSException) [pure virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions

CMSException (p. 1074) - If the provider fails to perform the reset operation.

MessageFormatException (p. 2493) - If the Message (p. 2375) has an invalid format.

6.170.3.18 virtual void cms::BytesMessage::setBodyBytes (const unsigned char * buffer, int numBytes) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]

sets the bytes given to the message body.

Parameters

buffer Byte Buffer to copy

numBytes Number of bytes in Buffer to copy

Exceptions

CMSException (p. 1074) - If an internal error occurs.

MessageNotWriteableException (p. 2549) - if in Read Only Mode.

6.170.3.19 `virtual void cms::BytesMessage::writeBoolean (bool value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a boolean to the bytes message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

value boolean to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.20 `virtual void cms::BytesMessage::writeByte (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a byte to the bytes message stream as a 1-byte value.

Parameters

value byte to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.21 `virtual void cms::BytesMessage::writeBytes (const std::vector< unsigned char > & value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

Parameters

value bytes to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.22 `virtual void cms::BytesMessage::writeBytes (const unsigned char * value, int offset, int length) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a portion of a byte array to the bytes message stream.

size as the number of bytes to write.

Parameters

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.23 `virtual void cms::BytesMessage::writeChar (char value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a char to the bytes message stream as a 1-byte value.

Parameters

value char to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.24 `virtual void cms::BytesMessage::writeDouble (double value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a double to the bytes message stream as a 8 byte value.

Parameters

value double to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.25 `virtual void cms::BytesMessage::writeFloat (float value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a float to the bytes message stream as a 4 byte value.

Parameters

value float to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.26 `virtual void cms::BytesMessage::writeInt (int value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a signed int to the bytes message stream as a 4 byte value.

Parameters

value signed int to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.27 `virtual void cms::BytesMessage::writeLong (long long value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a long long to the bytes message stream as a 8 byte value.

Parameters

value signed long long to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.28 `virtual void cms::BytesMessage::writeShort (short value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a signed short to the bytes message stream as a 2 byte value.

Parameters

value signed short to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.29 `virtual void cms::BytesMessage::writeString (const std::string & value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes an ASCII String to the Bytes message stream.

Parameters

value String to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.30 `virtual void cms::BytesMessage::writeUnsignedShort (unsigned short value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters

value unsigned short to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.170.3.31 `virtual void cms::BytesMessage::writeUTF (const std::string & value) throw (cms::MessageNotWriteableException, cms::CMSException)`
 [pure virtual]

Writes an UTF String to the **BytesMessage** (p. 979) stream.

Parameters

value String to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

The documentation for this class was generated from the following file:

- `src/main/cms/BytesMessage.h`

6.171 activemq::cmsutil::CachedConsumer Class Reference

A cached message consumer contained within a pooled session.

`#include <src/main/activemq/cmsutil/CachedConsumer.h>`

Inheritance diagram for `activemq::cmsutil::CachedConsumer`:

Public Member Functions

- **CachedConsumer** (`cms::MessageConsumer *consumer`)
- virtual `~CachedConsumer ()`
- virtual void **close** () throw (`cms::CMSException`)
Does nothing - the real producer resource will be closed by the lifecycle manager.
- virtual `cms::Message * receive` () throw (`cms::CMSException`)
Synchronously Receive a Message.
- virtual `cms::Message * receive` (int millisecs) throw (`cms::CMSException`)
Synchronously Receive a Message, time out after defined interval.
- virtual `cms::Message * receiveNoWait` () throw (`cms::CMSException`)
Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.
- virtual void **setMessageListener** (`cms::MessageListener *listener`) throw (`cms::CMSException`)
Sets the MessageListener that this class will send notifs on.

- virtual `cms::MessageListener * getMessageListener () const throw (cms::CMSEException)`
Gets the MessageListener that this class will send new Message notification events to.
- virtual `std::string getMessageSelector () const throw (cms::CMSEException)`
Gets this message consumer's message selector expression.

Protected Member Functions

- `CachedConsumer (const CachedConsumer &)`
- `CachedConsumer & operator= (const CachedConsumer &)`

6.171.1 Detailed Description

A cached message consumer contained within a pooled session.

6.171.2 Constructor & Destructor Documentation

- 6.171.2.1** `activemq::cmsutil::CachedConsumer::CachedConsumer (const CachedConsumer &) [inline, protected]`
- 6.171.2.2** `activemq::cmsutil::CachedConsumer::CachedConsumer (cms::MessageConsumer * consumer) [inline]`
- 6.171.2.3** `virtual activemq::cmsutil::CachedConsumer::~~CachedConsumer () [inline, virtual]`

6.171.3 Member Function Documentation

- 6.171.3.1** `virtual void activemq::cmsutil::CachedConsumer::close () throw (cms::CMSEException) [inline, virtual]`

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements `cms::Closeable` (p. 1065).

- 6.171.3.2** `virtual cms::MessageListener* activemq::cmsutil::CachedConsumer::getMessageListener () const throw (cms::CMSEException) [inline, virtual]`

Gets the MessageListener that this class will send new Message notification events to.

Returns

The listener of messages received by this consumer

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2424).

6.171.3.3 `virtual std::string activemq::cmsutil::CachedConsumer::getMessageSelector () const throw (cms::CMSException) [inline, virtual]`

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

CMSException - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2424).

6.171.3.4 `CachedConsumer& activemq::cmsutil::CachedConsumer::operator= (const CachedConsumer &) [inline, protected]`

6.171.3.5 `virtual cms::Message* activemq::cmsutil::CachedConsumer::receive () throw (cms::CMSException) [inline, virtual]`

Synchronously Receive a Message.

Returns

new message which the caller owns and must delete.

Exceptions

CMSException - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2425).

6.171.3.6 `virtual cms::Message* activemq::cmsutil::CachedConsumer::receive (int millisecs) throw (cms::CMSException) [inline, virtual]`

Synchronously Receive a Message, time out after defined interval.

Returns null if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

CMSException - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2425).

6.171.3.7 `virtual cms::Message* activemq::cmsutil::CachedConsumer::receiveNoWait ()
throw (cms::CMSEException) [inline, virtual]`

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2425).

6.171.3.8 `virtual void activemq::cmsutil::CachedConsumer::setMessageListener
(cms::MessageListener * listener) throw (cms::CMSEException)
[inline, virtual]`

Sets the MessageListener that this class will send notifs on.

Parameters

listener The listener of messages received by this consumer.

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2425).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CachedConsumer.h`

6.172 activemq::cmsutil::CachedProducer Class Reference

A cached message producer contained within a pooled session.

`#include <src/main/activemq/cmsutil/CachedProducer.h>`

Inheritance diagram for `activemq::cmsutil::CachedProducer`:

Public Member Functions

- `CachedProducer (cms::MessageProducer *producer)`
- `virtual ~CachedProducer ()`
- `virtual void close () throw (cms::CMSEException)`

Does nothing - the real producer resource will be closed by the lifecycle manager.

- virtual void **send** (**cms::Message** *message) throw (cms::CMSEException)
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (**cms::Message** *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException)
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message) throw (cms::CMSEException)
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException)
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **setDeliveryMode** (int mode) throw (cms::CMSEException)
Sets the delivery mode for this Producer.
- virtual int **getDeliveryMode** () const throw (cms::CMSEException)
Gets the delivery mode for this Producer.
- virtual void **setDisableMessageID** (bool value) throw (cms::CMSEException)
Sets if Message Ids are disabled for this Producer.
- virtual bool **getDisableMessageID** () const throw (cms::CMSEException)
Gets if Message Ids are disabled for this Producer.
- virtual void **setDisableMessageTimeStamp** (bool value) throw (cms::CMSEException)
Sets if Message Time Stamps are disabled for this Producer.
- virtual bool **getDisableMessageTimeStamp** () const throw (cms::CMSEException)
Gets if Message Time Stamps are disabled for this Producer.
- virtual void **setPriority** (int priority) throw (cms::CMSEException)
Sets the Priority that this Producers sends messages at.
- virtual int **getPriority** () const throw (cms::CMSEException)
Gets the Priority level that this producer sends messages at.
- virtual void **setTimeToLive** (long long time) throw (cms::CMSEException)
Sets the Time to Live that this Producers sends messages with.
- virtual long long **getTimeToLive** () const throw (cms::CMSEException)
Gets the Time to Live that this producer sends messages with.

Protected Member Functions

- **CachedProducer** (const **CachedProducer** &)
- **CachedProducer** & **operator=** (const **CachedProducer** &)

6.172.1 Detailed Description

A cached message producer contained within a pooled session.

6.172.2 Constructor & Destructor Documentation

6.172.2.1 **activemq::cmsutil::CachedProducer::CachedProducer** (const **CachedProducer** &) [inline, protected]

6.172.2.2 **activemq::cmsutil::CachedProducer::CachedProducer** (**cms::MessageProducer** * *producer*) [inline]

6.172.2.3 **virtual activemq::cmsutil::CachedProducer::~~CachedProducer** () [inline, virtual]

6.172.3 Member Function Documentation

6.172.3.1 **virtual void activemq::cmsutil::CachedProducer::close** () **throw** (**cms::CMSEException**) [inline, virtual]

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements **cms::Closeable** (p.1065).

6.172.3.2 **virtual int activemq::cmsutil::CachedProducer::getDeliveryMode** () **const throw** (**cms::CMSEException**) [inline, virtual]

Gets the delivery mode for this Producer.

Returns

The DeliveryMode

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p.2552).

6.172.3.3 **virtual bool activemq::cmsutil::CachedProducer::getDisableMessageID** () **const throw** (**cms::CMSEException**) [inline, virtual]

Gets if Message Ids are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2552).

6.172.3.4 `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageTimeStamp () const throw (cms::CMSEException) [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2553).

6.172.3.5 `virtual int activemq::cmsutil::CachedProducer::getPriority () const throw (cms::CMSEException) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2553).

6.172.3.6 `virtual long long activemq::cmsutil::CachedProducer::getTimeToLive () const throw (cms::CMSEException) [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

Returns

Time to live value in milliseconds

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2553).

6.172.3.7 `CachedProducer& activemq::cmsutil::CachedProducer::operator= (const CachedProducer &) [inline, protected]`

6.172.3.8 `virtual void activemq::cmsutil::CachedProducer::send (cms::Message * message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException) [inline, virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements `cms::MessageProducer` (p. 2553).

6.172.3.9 `virtual void activemq::cmsutil::CachedProducer::send (const cms::Destination * destination, cms::Message * message) throw (cms::CMSException) [inline, virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

destination The destination on which to send the message

message the message to be sent.

Exceptions

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements `cms::MessageProducer` (p. 2555).

6.172.3.10 `virtual void activemq::cmsutil::CachedProducer::send (cms::Message * message) throw (cms::CMSEException) [inline, virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

message The message to be sent.

Exceptions

CMSEException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements `cms::MessageProducer` (p. 2555).

6.172.3.11 `virtual void activemq::cmsutil::CachedProducer::send (const cms::Destination * destination, cms::Message * message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException) [inline, virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters

destination The destination on which to send the message

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSEException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements `cms::MessageProducer` (p. 2554).

6.172.3.12 `virtual void activemq::cmsutil::CachedProducer::setDeliveryMode (int mode) throw (cms::CMSEException) [inline, virtual]`

Sets the delivery mode for this Producer.

Parameters

mode The DeliveryMode

Exceptions

CMSEException - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2556).

6.172.3.13 `virtual void activemq::cmsutil::CachedProducer::setDisableMessageID (bool value) throw (cms::CMSEException) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

Parameters

value boolean indicating enable / disable (true / false)

Exceptions

CMSEException - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2556).

6.172.3.14 `virtual void activemq::cmsutil::CachedProducer::setDisableMessageTimeStamp (bool value) throw (cms::CMSEException) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

Parameters

value - boolean indicating enable / disable (true / false)

Exceptions

CMSEException - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2556).

6.172.3.15 `virtual void activemq::cmsutil::CachedProducer::setPriority (int priority) throw (cms::CMSEException) [inline, virtual]`

Sets the Priority that this Producers sends messages at.

Parameters

priority int value for Priority level

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2556).

6.172.3.16 **virtual void activemq::cmsutil::CachedProducer::setTimeToLive (long long *time*) throw (cms::CMSEException)** [inline, virtual]

Sets the Time to Live that this Producers sends messages with.

This value will be used if the time to live is not specified via the send method.

Parameters

time default time to live value in milliseconds

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2557).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CachedProducer.h**

6.173 decaf::util::concurrent::Callable< V > Class Template Reference

A task that returns a result and may throw an exception.

```
#include <src/main/decaf/util/concurrent/Callable.h>
```

Public Member Functions

- virtual **~Callable** ()
- virtual **V call** ()=0 throw (decaf::lang::Exception)
Computes a result, or throws an exception if unable to do so.

6.173.1 Detailed Description

```
template<typename V> class decaf::util::concurrent::Callable< V >
```

A task that returns a result and may throw an exception. Implementors define a single method with no arguments called call. This interface differs from the Runnable interface in that a **Callable** (p. 1003) object can return a result and is allowed to throw an exceptions from its call method.

The Executors class contains utility methods to convert from other common forms to **Callable** (p. 1003) classes.

Since

1.0

6.173.2 Constructor & Destructor Documentation

6.173.2.1 `template<typename V > virtual decaf::util::concurrent::Callable< V >::~~Callable () [inline, virtual]`

6.173.3 Member Function Documentation

6.173.3.1 `template<typename V > virtual V decaf::util::concurrent::Callable< V >::call () throw (decaf::lang::Exception) [pure virtual]`

Computes a result, or throws an exception if unable to do so.

Returns

Computed Result.

Exceptions

Exception If unable to compute a result.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Callable.h`

6.174 decaf::util::concurrent::CancellationException Class Reference

```
#include <src/main/decaf/util/concurrent/CancellationException.h>
```

Inheritance diagram for decaf::util::concurrent::CancellationException:

Public Member Functions

- **CancellationException** () throw ()
Default Constructor.
- **CancellationException** (const decaf::lang::Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CancellationException** (const CancellationException &ex) throw ()
Copy Constructor.
- **CancellationException** (const std::exception *cause) throw ()
Constructor.
- **CancellationException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.

- **CancellationException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **CancellationException** * clone () const

Clones this exception.

- virtual ~**CancellationException** () throw ()

6.174.1 Constructor & Destructor Documentation

6.174.1.1 decaf::util::concurrent::CancellationException::CancellationException () throw () [inline]

Default Constructor.

6.174.1.2 decaf::util::concurrent::CancellationException::CancellationException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.174.1.3 decaf::util::concurrent::CancellationException::CancellationException (const CancellationException & ex) throw () [inline]

Copy Constructor.

Parameters

ex - The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

6.174.1.4 decaf::util::concurrent::CancellationException::CancellationException (const std::exception * cause) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.174.1.5 `decaf::util::concurrent::CancellationException::CancellationException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

msg - The message to report

... - list of primitives that are formatted into the message

6.174.1.6 `decaf::util::concurrent::CancellationException::CancellationException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

cause - The exception that was the cause for this one to be thrown.

msg - The message to report

... - list of primitives that are formatted into the message

6.174.1.7 `virtual decaf::util::concurrent::CancellationException::~~CancellationException () throw ()` [inline, virtual]

6.174.2 Member Function Documentation

6.174.2.1 `virtual CancellationException* decaf::util::concurrent::CancellationException::clone () const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an exception that is a clone of this one.

Reimplemented from `decaf::lang::Exception` (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/CancellationException.h`

6.175 decaf::security::cert::Certificate Class Reference

Base interface for all identity certificates.

```
#include <src/main/decaf/security/cert/Certificate.h>
```

Inheritance diagram for decaf::security::cert::Certificate:

Public Member Functions

- virtual **~Certificate** ()
- virtual bool **equals** (const **Certificate** &cert) const =0
Compares the encoded form of the two certificates.
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)
Provides the encoded form of this certificate.
- virtual std::string **getType** () const =0
Returns the type of this certificate.
- virtual **PublicKey** * **getPublicKey** ()=0
Gets the public key of this certificate.
- virtual const **PublicKey** * **getPublicKey** () const =0
Gets the public key of this certificate.
- virtual void **verify** (const **PublicKey** &publicKey) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)
Verifies that this certificate was signed with the private key that corresponds to the specified public key.
- virtual void **verify** (const **PublicKey** &publicKey, const std::string &sigProvider) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)
Verifies that this certificate was signed with the private key that corresponds to the specified public key.
- virtual std::string **toString** () const =0
Returns a string representation of this certificate.

6.175.1 Detailed Description

Base interface for all identity certificates.

6.175.2 Constructor & Destructor Documentation

6.175.2.1 `virtual decaf::security::cert::Certificate::~~Certificate () [inline, virtual]`

6.175.3 Member Function Documentation

6.175.3.1 `virtual bool decaf::security::cert::Certificate::equals (const Certificate & cert) const [pure virtual]`

Compares the encoded form of the two certificates.

Parameters

cert The certificate to be tested for equality with this certificate.

Returns

true if the given certificate is equal to this certificate.

6.175.3.2 `virtual void decaf::security::cert::Certificate::getEncoded (std::vector< unsigned char > & output) const throw (CertificateEncodingException) [pure virtual]`

Provides the encoded form of this certificate.

Parameters

output Receives the encoded form of this certificate.

Exceptions

CertificateEncodingException (p. 1010) if an encoding error occurs

6.175.3.3 `virtual PublicKey* decaf::security::cert::Certificate::getPublicKey () [pure virtual]`

Gets the public key of this certificate.

Returns

the public key

6.175.3.4 `virtual const PublicKey* decaf::security::cert::Certificate::getPublicKey () const [pure virtual]`

Gets the public key of this certificate.

Returns

the public key

6.175.3.5 `virtual std::string decaf::security::cert::Certificate::getType () const`
`[pure virtual]`

Returns the type of this certificate.

Returns

the type of this certificate

6.175.3.6 `virtual std::string decaf::security::cert::Certificate::toString () const`
`[pure virtual]`

Returns a string representation of this certificate.

Returns

a string representation of this certificate

6.175.3.7 `virtual void decaf::security::cert::Certificate::verify (const`
`PublicKey & publicKey, const std::string & sigProvider)`
`const throw (NoSuchAlgorithmException, InvalidKeyException,`
`NoSuchProviderException, SignatureException, CertificateException)`
`[pure virtual]`

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Uses the verification engine of the specified provider.

Parameters

publicKey The public key used to carry out the validation.

sigProvider The name of the signature provider

Exceptions

NoSuchAlgorithmException (p. 2643) - on unsupported signature algorithms.

InvalidKeyException (p. 1994) - on incorrect key.

NoSuchProviderException (p. 2648) - if there's no default provider.

SignatureException (p. 3276) - on signature errors.

CertificateException (p. 1012) - on encoding errors.

6.175.3.8 `virtual void decaf::security::cert::Certificate::verify (const PublicKey`
`& publicKey) const throw (NoSuchAlgorithmException,`
`InvalidKeyException, NoSuchProviderException, SignatureException,`
`CertificateException) [pure virtual]`

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Parameters

publicKey The public key used to carry out the validation.

Exceptions

NoSuchAlgorithmException (p. 2643) - on unsupported signature algorithms.

InvalidKeyException (p. 1994) - on incorrect key.

NoSuchProviderException (p. 2648) - if there's no default provider.

SignatureException (p. 3276) - on signature errors.

CertificateException (p. 1012) - on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**Certificate.h**

6.176 decaf::security::cert::CertificateEncodingException Class Reference

```
#include <src/main/decaf/security/cert/CertificateEncodingException.h>
```

Inheritance diagram for decaf::security::cert::CertificateEncodingException:

Public Member Functions

- **CertificateEncodingException** () throw ()
Default Constructor.
- **CertificateEncodingException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateEncodingException** (const **CertificateEncodingException** &ex) throw ()
Copy Constructor.
- **CertificateEncodingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateEncodingException** * **clone** () const
Clones this exception.
- virtual ~**CertificateEncodingException** () throw ()

6.176.1 Constructor & Destructor Documentation

6.176.1.1 decaf::security::cert::CertificateEncodingException::CertificateEncodingException
() throw () [inline]

Default Constructor.

6.176.1.2 `decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.176.1.3 `decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const CertificateEncodingException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.176.1.4 `decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.176.1.5 `virtual decaf::security::cert::CertificateEncodingException::~~CertificateEncodingException () throw () [inline, virtual]`

6.176.2 Member Function Documentation

6.176.2.1 `virtual CertificateEncodingException* decaf::security::cert::CertificateEncodingException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p.1013).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateEncodingException.h`

6.177 decaf::security::cert::CertificateException Class Reference

```
#include <src/main/decaf/security/cert/CertificateException.h>
```

Inheritance diagram for `decaf::security::cert::CertificateException`:

Public Member Functions

- **CertificateException** () throw ()
Default Constructor.
- **CertificateException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateException** (const **CertificateException** &ex) throw ()
Copy Constructor.
- **CertificateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateException** * **clone** () const
Clones this exception.
- virtual ~**CertificateException** () throw ()

6.177.1 Constructor & Destructor Documentation

6.177.1.1 decaf::security::cert::CertificateException::CertificateException () throw () [inline]

Default Constructor.

6.177.1.2 decaf::security::cert::CertificateException::CertificateException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.177.1.3 `decaf::security::cert::CertificateException::CertificateException (const CertificateException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.177.1.4 `decaf::security::cert::CertificateException::CertificateException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.177.1.5 `virtual decaf::security::cert::CertificateException::~~CertificateException () throw () [inline, virtual]`

6.177.2 Member Function Documentation

6.177.2.1 `virtual CertificateException* decaf::security::cert::CertificateException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 1847).

Reimplemented in `decaf::security::cert::CertificateEncodingException` (p. 1011), `decaf::security::cert::CertificateExpiredException` (p. 1015), `decaf::security::cert::CertificateNotYetValidException` (p. 1017), and `decaf::security::cert::CertificateParsingException` (p. 1019).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateException.h`

6.178 decaf::security::cert::CertificateExpiredException Class Reference

```
#include <src/main/decaf/security/cert/CertificateExpiredException.h>
```

Inheritance diagram for decaf::security::cert::CertificateExpiredException:

Public Member Functions

- **CertificateExpiredException** () throw ()
Default Constructor.
- **CertificateExpiredException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateExpiredException** (const **CertificateExpiredException** &ex) throw ()
Copy Constructor.
- **CertificateExpiredException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateExpiredException** * clone () const
Clones this exception.
- virtual ~**CertificateExpiredException** () throw ()

6.178.1 Constructor & Destructor Documentation

6.178.1.1 decaf::security::cert::CertificateExpiredException::CertificateExpiredException
() throw () [inline]

Default Constructor.

6.178.1.2 decaf::security::cert::CertificateExpiredException::CertificateExpiredException
(const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.178.1.3 decaf::security::cert::CertificateExpiredException::CertificateExpiredException
(const CertificateExpiredException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.178.1.4 `decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.178.1.5 `virtual decaf::security::cert::CertificateExpiredException::~CertificateExpiredException () throw () [inline, virtual]`

6.178.2 Member Function Documentation

6.178.2.1 `virtual CertificateExpiredException* decaf::security::cert::CertificateExpiredException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.1013).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateExpiredException.h`

6.179 decaf::security::cert::CertificateNotYetValidException Class Reference

```
#include <src/main/decaf/security/cert/CertificateNotYetValidException.h>
```

Inheritance diagram for `decaf::security::cert::CertificateNotYetValidException`:

Public Member Functions

- **CertificateNotYetValidException** () throw ()
Default Constructor.
- **CertificateNotYetValidException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateNotYetValidException** (const **CertificateNotYetValidException** &ex) throw ()
Copy Constructor.
- **CertificateNotYetValidException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateNotYetValidException** * clone () const
Clones this exception.
- virtual ~**CertificateNotYetValidException** () throw ()

6.179.1 Constructor & Destructor Documentation

6.179.1.1 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException () throw () [inline]

Default Constructor.

6.179.1.2 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.179.1.3 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const CertificateNotYetValidException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.179.1.4 `decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.179.1.5 `virtual decaf::security::cert::CertificateNotYetValidException::~~CertificateNotYetValidException () throw () [inline, virtual]`

6.179.2 Member Function Documentation

6.179.2.1 `virtual CertificateNotYetValidException* decaf::security::cert::CertificateNotYetValidException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.1013).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateNotYetValidException.h`

6.180 decaf::security::cert::CertificateParsingException Class Reference

```
#include <src/main/decaf/security/cert/CertificateParsingException.h>
```

Inheritance diagram for `decaf::security::cert::CertificateParsingException`:

Public Member Functions

- **CertificateParsingException** () throw ()
Default Constructor.
- **CertificateParsingException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateParsingException** (const **CertificateParsingException** &ex) throw ()
Copy Constructor.
- **CertificateParsingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateParsingException** * clone () const
Clones this exception.
- virtual ~**CertificateParsingException** () throw ()

6.180.1 Constructor & Destructor Documentation

6.180.1.1 decaf::security::cert::CertificateParsingException::CertificateParsingException () throw () [inline]

Default Constructor.

6.180.1.2 decaf::security::cert::CertificateParsingException::CertificateParsingException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.180.1.3 decaf::security::cert::CertificateParsingException::CertificateParsingException (const CertificateParsingException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.180.1.4 `decaf::security::cert::CertificateParsingException::CertificateParsingException`
 (`const char * file`, `const int lineNumber`, `const char * msg`, ...)
 throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.180.1.5 `virtual`
`decaf::security::cert::CertificateParsingException::~~CertificateParsingException`
 () throw () [inline, virtual]

6.180.2 Member Function Documentation

6.180.2.1 `virtual CertificateParsingException* de-`
`caf::security::cert::CertificateParsingException::clone (`
`) const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.1013).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateParsingException.h`

6.181 decaf::lang::Character Class Reference

```
#include <src/main/decaf/lang/Character.h>
```

Inheritance diagram for `decaf::lang::Character`:

Public Member Functions

- `Character` (char value)

- virtual int **compareTo** (const **Character** &c) const
*Compares this **Character** (p. 1019) instance with another.*
- virtual bool **operator==** (const **Character** &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Character** &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const char &c) const
*Compares this **Character** (p. 1019) instance with a char type.*
- virtual bool **operator==** (const char &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const char &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const **Character** &c) const
- bool **equals** (const char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static **Character** **valueOf** (char value)
*Returns a **Character** (p. 1019) instance representing the specified char value.*
- static bool **isWhitespace** (char c)
Indicates whether or not the given character is considered whitespace.

- static bool **isDigit** (char c)
Indicates whether or not the given character is a digit.
- static bool **isLowerCase** (char c)
Indicates whether or not the given character is a lower case character.
- static bool **isUpperCase** (char c)
Indicates whether or not the given character is a upper case character.
- static bool **isLetter** (char c)
Indicates whether or not the given character is a letter.
- static bool **isLetterOrDigit** (char c)
Indicates whether or not the given character is either a letter or a digit.
- static bool **isISOControl** (char c)
Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.
- static int **digit** (char c, int radix)
Returns the numeric value of the character ch in the specified radix.

Static Public Attributes

- static const int **MIN_RADIX** = 2
The minimum radix available for conversion to and from strings.
- static const int **MAX_RADIX** = 36
The maximum radix available for conversion to and from strings.
- static const char **MIN_VALUE** = (char)0x7F
The minimum value that a signed char can take on.
- static const char **MAX_VALUE** = (char)0x80
The maximum value that a signed char can take on.
- static const int **SIZE** = 8
The size of the primitive character in bits.

6.181.1 Constructor & Destructor Documentation

6.181.1.1 decaf::lang::Character::Character (char value)

Parameters

value - char to wrap.

6.181.2 Member Function Documentation

6.181.2.1 virtual unsigned char decaf::lang::Character::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

6.181.2.2 virtual int decaf::lang::Character::compareTo (const char & c) const [inline, virtual]

Compares this **Character** (p. 1019) instance with a char type.

Parameters

c - the char instance to be compared

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **char** > (p. 1126).

6.181.2.3 virtual int decaf::lang::Character::compareTo (const Character & c) const [inline, virtual]

Compares this **Character** (p. 1019) instance with another.

Parameters

c - the **Character** (p. 1019) instance to be compared

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.181.2.4 static int decaf::lang::Character::digit (char c, int radix) [static]

Returns the numeric value of the character *ch* in the specified radix.

If the radix is not in the range `MIN_RADIX <= radix <= MAX_RADIX` or if the value of *ch* is not a valid digit in the specified radix, -1 is returned. A character is a valid digit if at least one of the following is true:

* The method `isDigit` is true of the character and the single-character decomposition is less than the specified radix. In this case the decimal digit value is returned. * The character is one of the

uppercase Latin letters 'A' through 'Z' and its code is less than $\text{radix} + 'A' - 10$. In this case, $\text{ch} - 'A' + 10$ is returned. * The character is one of the lowercase Latin letters 'a' through 'z' and its code is less than $\text{radix} + 'a' - 10$. In this case, $\text{ch} - 'a' + 10$ is returned.

Parameters

c - the char to be converted
radix - the radix of the number

Returns

the numeric value of the number represented in the given radix

6.181.2.5 `virtual double decaf::lang::Character::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.181.2.6 `bool decaf::lang::Character::equals (const char & c) const [inline, virtual]`

Returns

true if the two **Character** have the same value.

Implements **decaf::lang::Comparable**< **char** > (p.1126).

6.181.2.7 `bool decaf::lang::Character::equals (const Character & c) const [inline]`

Returns

true if the two **Character** (p.1019) Objects have the same value.

6.181.2.8 `virtual float decaf::lang::Character::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.181.2.9 `virtual int decaf::lang::Character::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.181.2.10 `static bool decaf::lang::Character::isDigit (char c) [inline, static]`

Indicates whether or not the given character is a digit.

6.181.2.11 `static bool decaf::lang::Character::isISOControl (char c) [inline, static]`

Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.

Parameters

`c` - the character, including supplementary characters

Returns

true if the char is an ISO control character

6.181.2.12 `static bool decaf::lang::Character::isLetter (char c) [inline, static]`

Indicates whether or not the given character is a letter.

6.181.2.13 `static bool decaf::lang::Character::isLetterOrDigit (char c) [inline, static]`

Indicates whether or not the given character is either a letter or a digit.

6.181.2.14 `static bool decaf::lang::Character::isLowerCase (char c) [inline, static]`

Indicates whether or not the given character is a lower case character.

6.181.2.15 `static bool decaf::lang::Character::isUpperCase (char c) [inline, static]`

Indicates whether or not the given character is a upper case character.

6.181.2.16 `static bool decaf::lang::Character::isWhitespace (char c) [inline, static]`

Indicates whether or not the given character is considered whitespace.

6.181.2.17 `virtual long long decaf::lang::Character::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.181.2.18 `virtual bool decaf::lang::Character::operator< (const Character & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.181.2.19 `virtual bool decaf::lang::Character::operator< (const char & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< char >` (p. 1127).

6.181.2.20 `virtual bool decaf::lang::Character::operator==(const Character & c) const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.181.2.21 `virtual bool decaf::lang::Character::operator==(const char & c) const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **char** > (p. 1127).

6.181.2.22 **virtual short decaf::lang::Character::shortValue () const** [inline, virtual]

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

6.181.2.23 **std::string decaf::lang::Character::toString () const**

Returns

this **Character** (p. 1019) Object as a **String** (p. 3427) Representation

6.181.2.24 **static Character decaf::lang::Character::valueOf (char value)**
[inline, static]

Returns a **Character** (p. 1019) instance representing the specified char value.

Parameters

value - the primitive char to wrap.

Returns

a new Charactor instance that wraps this value.

6.181.3 Field Documentation

6.181.3.1 **const int decaf::lang::Character::MAX_RADIX = 36** [static]

The maximum radix available for conversion to and from strings.

6.181.3.2 **const char decaf::lang::Character::MAX_VALUE = (char)0x80** [static]

The maximum value that a signed char can take on.

6.181.3.3 **const int decaf::lang::Character::MIN_RADIX = 2** [static]

The minimum radix available for conversion to and from strings.

6.181.3.4 **const char decaf::lang::Character::MIN_VALUE = (char)0x7F** [static]

The minimum value that a signed char can take on.

6.181.3.5 `const int decaf::lang::Character::SIZE = 8` [static]

The size of the primitive character in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Character.h`

6.182 `decaf::internal::nio::CharArrayBuffer` Class Reference

```
#include <src/main/decaf/internal/nio/CharArrayBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::CharArrayBuffer`:

Public Member Functions

- **CharArrayBuffer** (int size, bool **readOnly**=false) throw (decaf::lang::exceptions::IllegalArgumentException)

*Creates a **CharArrayBuffer** (p. 1027) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **CharArrayBuffer** (char *array, int size, int **offset**, int **length**, bool **readOnly**=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

*Creates a **CharArrayBuffer** (p. 1027) object that wraps the given array.*

- **CharArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int **offset**, int **length**, bool **readOnly**=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

*Creates a byte buffer that wraps the passed **ByteArrayAdapter** and start at the given offset.*

- **CharArrayBuffer** (const **CharArrayBuffer** &other)

*Create a **CharArrayBuffer** (p. 1027) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayAdapter** and when changes are made to that data it is reflected in both.*

- virtual ~**CharArrayBuffer** ()
- virtual char * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

*the array that backs this **Buffer** (p. 855).*

Exceptions

***ReadOnlyBufferException** (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.*

- virtual int **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.

- virtual CharBuffer * **asReadOnlyBuffer** () const

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only char buffer which the caller then owns.

- virtual CharBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n + 1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **CharBuffer** (p. 1037).

Exceptions

ReadOnlyBufferException (p. 2966) - If this buffer is read-only

- virtual CharBuffer * **duplicate** ()

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new char **Buffer** (p. 855) which the caller owns.

- virtual char **get** () throw (decaf::nio::BufferUnderflowException)

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

Returns

the char at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return

- virtual char **get** (int index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

Reads the char at the given index.

Parameters

index *The index in the **Buffer** (p. 855) where the char is to be read.*

Returns

the char that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit or is negative.

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

- virtual bool **isReadOnly** () const

- virtual CharBuffer & **put** (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes the given char into this buffer at the current position, and then increments the position.

Parameters

value *The char value to be written.*

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than ^{its limit}
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual CharBuffer & **put** (int index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes the given char into this buffer at the given index.

Parameters

index *The position in the **Buffer** (p. 855) to write the data.*
value *The char to write.*

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual CharBuffer * **slice** () const

Creates a new **CharBuffer** (p. 1037) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **CharBuffer** (p. 1037) which the caller owns.

- virtual lang::CharSequence * **subSequence** (int start, int end) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 860) + start, and its limit will be **position()** (p. 860) + end. The new **Buffer** (p. 855) will be read-only if, and only if, this buffer is read-only.

Parameters

start The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 860).

end The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than **remaining()** (p. 860).

Returns

The new character buffer, caller owns.

Exceptions

IndexOutOfBoundsException if the preconditions on start and end fail.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **CharArrayBuffer** (p. 1027) as Read-Only.

Protected Attributes

- bool **readOnly**
- decaf::lang::Pointer< ByteArrayAdapter > **__array**
- int **offset**
- int **length**

6.182.1 Constructor & Destructor Documentation

6.182.1.1 `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (int size, bool readOnly = false) throw (decaf::lang::exceptions::IllegalArgumentException)`

Creates a **CharArrayBuffer** (p. 1027) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

size The size of the array, this is the limit we read and write to.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArgumentException if the capacity value is negative.

6.182.1.2 `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (char * array, int size, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a **CharArrayBuffer** (p. 1027) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The actual array to wrap.

size The size of the given array.

offset The position that is this buffers start position.

length The limit of how many bytes into the array this Buffer can write.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException if buffer is NULL

IndexOutOfBoundsException if offset is greater than array capacity.

6.182.1.3 `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **CharArrayBuffer** (p. 1027) will be that of the remaining capacity of the passed buffer.

Parameters

- array* The ByteArrayAdapter to wrap.
- offset* The position that is this buffers start position.
- length* The limit of how many bytes into the array this Buffer can write.
- readOnly* Boolean indicating if this buffer should be read-only, default as false.

Exceptions

- NullPointerException* if array is NULL
- IndexOutOfBoundsException* if offset + length is greater than array size.

6.182.1.4 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (const CharArrayBuffer & *other*)

Create a **CharArrayBuffer** (p. 1027) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

Parameters

- other* The **CharArrayBuffer** (p. 1027) this one is to mirror.

6.182.1.5 virtual decaf::internal::nio::CharArrayBuffer::~~CharArrayBuffer () [virtual]

6.182.2 Member Function Documentation

6.182.2.1 virtual char* decaf::internal::nio::CharArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

- the array that backs this **Buffer** (p. 855).

Exceptions

- ReadOnlyBufferException* (p. 2966) if this **Buffer** (p. 855) is read only.
- UnsupportedOperationException* if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 1043).

6.182.2.2 `virtual int decaf::internal::nio::CharArrayBuffer::arrayOffset ()
throw (decaf::lang::exceptions::UnsupportedOperationException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 1043).

6.182.2.3 `virtual CharBuffer* de-
caf::internal::nio::CharArrayBuffer::asReadOnlyBuffer ()
const [virtual]`

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only char buffer which the caller then owns.

Implements **decaf::nio::CharBuffer** (p. 1043).

6.182.2.4 `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::compact ()
throw (decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 860) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 859) - 1 is copied to index `n = limit()` (p. 859) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **CharBuffer** (p. 1037).

Exceptions

ReadOnlyBufferException (p. 2966) - If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 1044).

6.182.2.5 `virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::duplicate ()`
[virtual]

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new char **Buffer** (p. 855) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 1044).

6.182.2.6 `virtual char decaf::internal::nio::CharArrayBuffer::get () throw (`
`decaf::nio::BufferUnderflowException)` [virtual]

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

Returns

the char at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return

Implements **decaf::nio::CharBuffer** (p. 1045).

6.182.2.7 `virtual char decaf::internal::nio::CharArrayBuffer::get (int index)`
`const throw (lang::exceptions::IndexOutOfBoundsException)` [virtual]

Absolute get method.

Reads the char at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the char is to be read.

Returns

the char that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit or is negative.

Implements **decaf::nio::CharBuffer** (p. 1045).

6.182.2.8 `virtual bool decaf::internal::nio::CharArrayBuffer::hasArray () const`
[inline, virtual]

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::CharBuffer** (p. 1047).

6.182.2.9 `virtual bool decaf::internal::nio::CharArrayBuffer::isReadOnly () const`
[inline, virtual]

6.182.2.10 `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put`
`(char value) throw (decaf::nio::BufferOverflowException,`
`decaf::nio::ReadOnlyBufferException)` [virtual]

Writes the given char into this buffer at the current position, and then increments the position.

Parameters

value The char value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 1048).

6.182.2.11 `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put`
`(int index, char value) throw (de-`
`cafe::lang::exceptions::IndexOutOfBoundsException,`
`decaf::nio::ReadOnlyBufferException)` [virtual]

Writes the given char into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The char to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 1049).

6.182.2.12 `virtual void decaf::internal::nio::CharArrayBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **CharArrayBuffer** (p. 1027) as Read-Only.

Parameters

value Boolean value, true if this buffer is to be read-only, false otherwise.

6.182.2.13 `virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::slice () const [virtual]`

Creates a new **CharBuffer** (p. 1037) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **CharBuffer** (p. 1037) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 1051).

6.182.2.14 `virtual lang::CharSequence* decaf::internal::nio::CharArrayBuffer::subSequence (int start, int end) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be

that of this buffer, its position will be **position()** (p. 860) + start, and its limit will be **position()** (p. 860) + end. The new **Buffer** (p. 855) will be read-only if, and only if, this buffer is read-only.

Parameters

- start** The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 860).
- end** The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than **remaining()** (p. 860).

Returns

The new character buffer, caller owns.

Exceptions

IndexOutOfBoundsException if the preconditions on start and end fail.

Implements **decaf::nio::CharBuffer** (p. 1051).

6.182.3 Field Documentation

- 6.182.3.1** **decaf::lang::Pointer<ByteArrayAdapter>**
decaf::internal::nio::CharArrayBuffer::_array [protected]
- 6.182.3.2** **int decaf::internal::nio::CharArrayBuffer::length** [protected]
- 6.182.3.3** **int decaf::internal::nio::CharArrayBuffer::offset** [protected]
- 6.182.3.4** **bool decaf::internal::nio::CharArrayBuffer::readOnly** [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/CharArrayBuffer.h`

6.183 decaf::nio::CharBuffer Class Reference

This class defines four categories of operations upon character buffers:

```
#include <src/main/decaf/nio/CharBuffer.h>
```

Inheritance diagram for **decaf::nio::CharBuffer**:

Public Member Functions

- **virtual ~CharBuffer ()**
- **virtual std::string toString () const**
- **CharBuffer & append (char value) throw (BufferOverflowException, ReadOnlyBufferException)**

Appends the specified character to this buffer.

- **CharBuffer** & **append** (const **lang::CharSequence** *value) throw (**BufferOverflowException**, **ReadOnlyBufferException**)

Appends the specified character sequence to this buffer.

- **CharBuffer** & **append** (const **lang::CharSequence** *value, int start, int end) throw (**decaf::lang::exceptions::IndexOutOfBoundsException**, **BufferOverflowException**, **ReadOnlyBufferException**)

Appends a subsequence of the specified character sequence to this buffer If value is Null the the string "null" is appended to the buffer.

- virtual char * **array** ()=0 throw (**decaf::lang::exceptions::UnsupportedOperationException**, **ReadOnlyBufferException**)

Returns the character array that backs this buffer (optional operation).

- virtual int **arrayOffset** ()=0 throw (**decaf::lang::exceptions::UnsupportedOperationException**, **ReadOnlyBufferException**)

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual **CharBuffer** * **asReadOnlyBuffer** () const =0

Creates a new, read-only char buffer that shares this buffer's content.

- char **charAt** (int index) const throw (**decaf::lang::exceptions::IndexOutOfBoundsException**)

Reads the character at the given index relative to the current position.

- virtual **CharBuffer** & **compact** ()=0 throw (**ReadOnlyBufferException**)

Compacts this buffer.

- virtual **CharBuffer** * **duplicate** ()=0

Creates a new char buffer that shares this buffer's content.

- virtual char **get** ()=0 throw (**BufferUnderflowException**)

Relative get method.

- virtual char **get** (int index) const =0 throw (**decaf::lang::exceptions::IndexOutOfBoundsException**)

Absolute get method.

- **CharBuffer** & **get** (std::vector< char > buffer) throw (**BufferUnderflowException**)

Relative bulk get method.

- **CharBuffer** & **get** (char *buffer, int size, int offset, int length) throw (**BufferUnderflowException**, **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IndexOutOfBoundsException**)

Relative bulk get method.

- virtual bool **hasArray** () const =0

Tells whether or not this buffer is backed by an accessible char array.

- **int length () const**
Returns the length of this character buffer.
- **CharBuffer & put (CharBuffer &src) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException)**
This method transfers the chars remaining in the given source buffer into this buffer.
- **CharBuffer & put (const char *buffer, int size, int offset, int length) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)**
This method transfers chars into this buffer from the given source array.
- **CharBuffer & put (std::vector< char > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)**
This method transfers the entire content of the given source char array into this buffer.
- **virtual CharBuffer & put (char value)=0 throw (BufferOverflowException, ReadOnlyBufferException)**
Writes the given char into this buffer at the current position, and then increments the position.
- **virtual CharBuffer & put (int index, char value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)**
Writes the given char into this buffer at the given index.
- **CharBuffer & put (std::string &src, int start, int end) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException)**
Relative bulk put method (optional operation).
- **CharBuffer & put (const std::string &src) throw (BufferOverflowException, ReadOnlyBufferException)**
Relative bulk put method (optional operation).
- **virtual int read (CharBuffer *target) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, ReadOnlyBufferException)**
Attempts to read characters into the specified character buffer.
- **virtual lang::CharSequence * subSequence (int start, int end) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)**
Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.
- **virtual CharBuffer * slice () const =0**
Creates a new CharBuffer (p. 1037) whose content is a shared subsequence of this buffer's content.
- **virtual int compareTo (const CharBuffer &value) const**

- virtual bool **equals** (const **CharBuffer** &value) const
- virtual bool **operator==** (const **CharBuffer** &value) const
- virtual bool **operator<** (const **CharBuffer** &value) const

Static Public Member Functions

- static **CharBuffer** * **allocate** (int capacity) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Allocates a new character buffer.
- static **CharBuffer** * **wrap** (char *array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
*Wraps the passed buffer with a new **CharBuffer** (p. 1037).*
- static **CharBuffer** * **wrap** (std::vector< char > &buffer)
*Wraps the passed STL char Vector in a **CharBuffer** (p. 1037).*

Protected Member Functions

- **CharBuffer** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)
*Creates a **CharBuffer** (p. 1037) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.183.1 Detailed Description

This class defines four categories of operations upon character buffers: o Absolute and relative get and put methods that read and write single characters; o Relative bulk get methods that transfer contiguous sequences of characters from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of characters from a character array, a string, or some other character buffer into this buffer. o Methods for compacting, duplicating, and slicing a character buffer.

Character buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing character array or string into a buffer, or by creating a view of an existing byte buffer

This class implements the CharSequence interface so that character buffers may be used wherever character sequences are accepted, for example in the regular-expression package decaf.util.regex.

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained. The sequence of statements

```
cb.put("text/"); cb.put(subtype); cb.put("; charset="); cb.put(enc);
```

can, for example, be replaced by the single statement

```
cb.put("text/").put(subtype).put("; charset=").put(enc);
```

6.183.2 Constructor & Destructor Documentation

6.183.2.1 `decaf::nio::CharBuffer::CharBuffer (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)` [protected]

Creates a **CharBuffer** (p. 1037) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity The size of the array, this is the limit we read and write to.

Exceptions

IllegalArgumentException if capacity is negative.

6.183.2.2 `virtual decaf::nio::CharBuffer::~~CharBuffer ()` [inline, virtual]

6.183.3 Member Function Documentation

6.183.3.1 `static CharBuffer* decaf::nio::CharBuffer::allocate (int capacity) throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [static]

Allocates a new character buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity The size of the Char buffer in chars (1 byte).

Returns

the **CharBuffer** (p. 1037) that was allocated, caller owns.

Exceptions

IndexOutOfBoundsException if capacity is negative.

6.183.3.2 `CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence * value, int start, int end) throw (decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException)`

Appends a subsequence of the specified character sequence to this buffer. If value is Null the the string "null" is appended to the buffer.

Parameters

value The CharSequence to append.
start The index to start appending from.
end The index to append to.

Returns

a reference to this modified **CharBuffer** (p. 1037).

Exceptions

BufferOverflowException (p. 880) if there is no more space
ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.
IndexOutOfBoundsException if start > end, or > length of sequence.

6.183.3.3 CharBuffer& decaf::nio::CharBuffer::append (char *value*) throw (BufferOverflowException, ReadOnlyBufferException)

Appends the specified character to this buffer.

Parameters

value The char to append.

Returns

a reference to this modified **CharBuffer** (p. 1037).

Exceptions

BufferOverflowException (p. 880) if there is no more space
ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

6.183.3.4 CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence * *value*) throw (BufferOverflowException, ReadOnlyBufferException)

Appends the specified character sequence to this buffer.

If value is Null the the string "null" is appended to the buffer.

Parameters

value The CharSequence to append.

Returns

a reference to this modified **CharBuffer** (p. 1037)

Exceptions

BufferOverflowException (p. 880) if there is no more space
ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

6.183.3.5 `virtual char* decaf::nio::CharBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 855).

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in `decaf::internal::nio::CharArrayBuffer` (p. 1032).

6.183.3.6 `virtual int decaf::nio::CharBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in `decaf::internal::nio::CharArrayBuffer` (p. 1033).

6.183.3.7 `virtual CharBuffer* decaf::nio::CharBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only char buffer which the caller then owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1033).

6.183.3.8 char decaf::nio::CharBuffer::charAt (int *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Reads the character at the given index relative to the current position.

Parameters

index - The index of the character to be read relative to position

Returns

The character at index **position()** (p. 860) + index.

Exceptions

IndexOutOfBoundsException if the index + the current position exceeds the size of the buffer or the index is negative.

6.183.3.9 virtual CharBuffer& decaf::nio::CharBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \mathbf{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index **limit()** (p. 859) - 1 is copied to index $n = \mathbf{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **CharBuffer** (p. 1037).

Exceptions

ReadOnlyBufferException (p. 2966) - If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1033).

6.183.3.10 virtual int decaf::nio::CharBuffer::compareTo (const CharBuffer & *value*) const [virtual]

6.183.3.11 virtual CharBuffer* decaf::nio::CharBuffer::duplicate () [pure virtual]

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new char **Buffer** (p. 855) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1034).

6.183.3.12 `virtual bool decaf::nio::CharBuffer::equals (const CharBuffer & value) const` [virtual]

6.183.3.13 `virtual char decaf::nio::CharBuffer::get () throw (BufferUnderflowException)` [pure virtual]

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

Returns

the char at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1034).

6.183.3.14 `virtual char decaf::nio::CharBuffer::get (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Absolute get method.

Reads the char at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the char is to be read.

Returns

the char that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit or is negative.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1034).

6.183.3.15 CharBuffer& decaf::nio::CharBuffer::get (std::vector< char > *buffer*) throw (BufferUnderflowException)

Relative bulk get method.

This method transfers chars from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **CharBuffer** (p. 1037).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than `length` chars remaining in this buffer.

6.183.3.16 CharBuffer& decaf::nio::CharBuffer::get (char * *buffer*, int *size*, int *offset*, int *length*) throw (BufferUnderflowException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Relative bulk get method.

This method transfers chars from this buffer into the given destination array. If there are fewer chars remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 860), then no bytes are transferred and a **BufferUnderflowException** (p. 882) is thrown.

Otherwise, this method copies `length` chars from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters

buffer The pointer to an allocated buffer to fill.

size The size of the buffer passed.

offset The position in the buffer to start filling.

length The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than `length` chars remaining in this buffer

NullPointerException if the passed buffer is null.

IndexOutOfBoundsException if the preconditions of `size`, `offset`, or `length` are not met.

6.183.3.17 `virtual bool decaf::nio::CharBuffer::hasArray () const` [pure virtual]

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1035).

6.183.3.18 `int decaf::nio::CharBuffer::length () const` [inline]

Returns the length of this character buffer.

Returns

the length of this buffer from the position to the limit.

6.183.3.19 `virtual bool decaf::nio::CharBuffer::operator< (const CharBuffer & value) const` [virtual]**6.183.3.20** `virtual bool decaf::nio::CharBuffer::operator== (const CharBuffer & value) const` [virtual]**6.183.3.21** `CharBuffer& decaf::nio::CharBuffer::put (CharBuffer & src) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException)`

This method transfers the chars remaining in the given source buffer into this buffer.

If there are more chars remaining in the source buffer than in this buffer, that is, if src.remaining() > remaining() (p. 860), then no chars are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies n = src.remaining() chars from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n.

Parameters

src - the buffer to take chars from an place in this one.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer for the remaining chars in the source buffer.

IllegalArgumentException if the source buffer is this buffer.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

6.183.3.22 CharBuffer& decaf::nio::CharBuffer::put (const char
* *buffer*, int *size*, int *offset*, int *length*) throw
(BufferOverflowException, ReadOnlyBufferException,
decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::NullPointerException)

This method transfers chars into this buffer from the given source array.

If there are more chars to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 860), then no chars are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

buffer The array from which chars are to be read.

size The size of the buffer passed.

offset The offset within the array of the first char to be read.

length The number of chars to be read from the given array.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer

ReadOnlyBufferException (p. 2966) if this buffer is read-only

NullPointerException if the passed buffer is null.

IndexOutOfBoundsException if the preconditions of `size`, `offset`, or `length` are not met.

6.183.3.23 virtual CharBuffer& decaf::nio::CharBuffer::put (char *value*) throw (**BufferOverflowException**, **ReadOnlyBufferException**) [pure virtual]

Writes the given char into this buffer at the current position, and then increments the position.

Parameters

value The char value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1035).

6.183.3.24 `virtual CharBuffer& decaf::nio::CharBuffer::put (int index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given char into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.
value The char to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1035).

6.183.3.25 `CharBuffer& decaf::nio::CharBuffer::put (std::string & src, int start, int end) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Relative bulk put method (optional operation).

This method transfers characters from the given string into this buffer. If there are more characters to be copied from the string than remain in this buffer, that is, if `end - start > remaining()` (p. 860), then no characters are transferred and a **BufferOverflowException** (p. 880) is thrown.

Returns

a reference to this buffer

Otherwise, this method copies `n = end - start` characters from the given string into this buffer, starting at the given start index and at the current position of this buffer. The position of this buffer is then incremented by `n`.

Parameters

src The string to copy from.
start The position in *src* to start from.
end The position in *src* to stop at.

Returns

a reference to this **CharBuffer** (p. 1037).

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only

6.183.3.26 CharBuffer& decaf::nio::CharBuffer::put (const std::string & *src*) throw (BufferOverflowException, ReadOnlyBufferException)

Relative bulk put method (optional operation).

This method transfers the entire content of the given source string into this buffer. An invocation of this method of the form `dst.put(s)` behaves in exactly the same way as the invocation.

Parameters

src The string to copy from.

Returns

a reference to this **CharBuffer** (p. 1037).

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

6.183.3.27 CharBuffer& decaf::nio::CharBuffer::put (std::vector< char > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source char array into this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size()`.

Parameters

buffer The buffer whose contents are copied to this **CharBuffer** (p. 1037).

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

6.183.3.28 virtual int decaf::nio::CharBuffer::read (CharBuffer * *target*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, ReadOnlyBufferException) [virtual]

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

Parameters

target The buffer to read characters into

Returns

The number of characters added to the buffer, or `string::npos` if this source of characters is at its end

Exceptions

NullPointerException if target is Null.

IllegalArgumentException if target is this **CharBuffer** (p. 1037).

ReadOnlyBufferException (p. 2966) if this buffer is in read-only mode.

6.183.3.29 `virtual CharBuffer* decaf::nio::CharBuffer::slice () const` [pure virtual]

Creates a new **CharBuffer** (p. 1037) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **CharBuffer** (p. 1037) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1036).

6.183.3.30 `virtual lang::CharSequence* decaf::nio::CharBuffer::subSequence (int start, int end) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 860) + start, and its limit will be **position()** (p. 860) + end. The new **Buffer** (p. 855) will be read-only if, and only if, this buffer is read-only.

Parameters

start The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 860).

end The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than **remaining()** (p. 860).

Returns

The new character buffer, caller owns.

Exceptions

IndexOutOfBoundsException if the preconditions on start and end fail.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1036).

6.183.3.31 virtual std::string decaf::nio::CharBuffer::toString () const [virtual]

Returns

a std::string describing this object

6.183.3.32 static CharBuffer* decaf::nio::CharBuffer::wrap (char * array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [static]

Wraps the passed buffer with a new **CharBuffer** (p. 1037).

The new buffer will be backed by the given char array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array The array that will back the new buffer.
size The size of the array passed in.
offset The offset of the subarray to be used.
length The length of the subarray to be used.

Returns

a new **CharBuffer** (p. 1037) that is backed by buffer, caller owns.

Exceptions

NullPointerException if the array pointer is Null.

IndexOutOfBoundsException if capacity is negative.

6.183.3.33 static CharBuffer* decaf::nio::CharBuffer::wrap (std::vector< char > & buffer) [static]

Wraps the passed STL char Vector in a **CharBuffer** (p. 1037).

The new buffer will be backed by the given char array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns

- a new **CharBuffer** (p. 1037) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**CharBuffer.h**

6.184 decaf::lang::CharSequence Class Reference

A **CharSequence** (p. 1053) is a readable sequence of char values.

```
#include <src/main/decaf/lang/CharSequence.h>
```

Inheritance diagram for decaf::lang::CharSequence:

Public Member Functions

- virtual **~CharSequence** ()
- virtual int **length** () const =0
- virtual char **charAt** (int index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

- virtual **CharSequence** * **subSequence** (int start, int end) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

*Returns a new **CharSequence** (p. 1053) that is a subsequence of this sequence.*

- virtual std::string **toString** () const =0

6.184.1 Detailed Description

A **CharSequence** (p. 1053) is a readable sequence of char values. This interface provides uniform, read-only access to many different kinds of char sequences.

This interface does not define that a **CharSequence** (p. 1053) should implement comparable, it is therefore up to the dervied classes that implement this interface to define equality, which implies that comparison of two CharSequences does not have a contract on equality.

6.184.2 Constructor & Destructor Documentation

6.184.2.1 `virtual decaf::lang::CharSequence::~~CharSequence () [inline, virtual]`

6.184.3 Member Function Documentation

6.184.3.1 `virtual char decaf::lang::CharSequence::charAt (int index) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

Parameters

index - position to return the char at.

Returns

the char at the given position

Exceptions

IndexOutOfBoundsException if index is > than `length()` (p. 1054) or negative

Implemented in `decaf::lang::String` (p. 3429).

6.184.3.2 `virtual int decaf::lang::CharSequence::length () const [pure virtual]`

Returns

the length of the underlying character sequence.

Implemented in `decaf::lang::String` (p. 3429).

6.184.3.3 `virtual CharSequence* decaf::lang::CharSequence::subSequence (int start, int end) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Returns a new **CharSequence** (p. 1053) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index `end - 1`. The length (in chars) of the returned sequence is `end - start`, so if `start == end` then an empty sequence is returned.

Parameters

start - the start index, inclusive

end - the end index, exclusive

Returns

a new **CharSequence** (p. 1053)

Exceptions

IndexOutOfBoundsException if start or end > **length()** (p.1054) or start or end are negative.

Implemented in **decaf::lang::String** (p. 3430).

6.184.3.4 **virtual std::string decaf::lang::CharSequence::toString () const** [pure virtual]

Returns

the string representation of this **CharSequence** (p. 1053)

Implemented in **decaf::lang::String** (p. 3430).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**CharSequence.h**

6.185 decaf::util::zip::CheckedInputStream Class Reference

An implementation of a **FilterInputStream** that will maintain a **Checksum** (p. 1059) of the bytes read, the **Checksum** (p. 1059) can then be used to verify the integrity of the input stream.

#include <src/main/decaf/util/zip/CheckedInputStream.h>

Inheritance diagram for decaf::util::zip::CheckedInputStream:

Public Member Functions

- **CheckedInputStream** (**InputStream** *inputStream, **Checksum** *sum, bool own=false)

*Create a new instance of a **CheckedInputStream** (p. 1055).*

- virtual ~**CheckedInputStream** ()
- **Checksum** * getChecksum () const

*Returns a Pointer to the **Checksum** (p. 1059) that is in use by this **CheckedInputStream** (p. 1055).*

- virtual long long **skip** (long long num) throw (decaf::io::IOException)

Skips over and discards n bytes of data from this input stream.

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

Protected Member Functions

- virtual int **doReadByte** () throw (decaf::io::IOException)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

6.185.1 Detailed Description

An implementation of a `FilterInputStream` that will maintain a **Checksum** (p. 1059) of the bytes read, the **Checksum** (p. 1059) can then be used to verify the integrity of the input stream.

Since

1.0

6.185.2 Constructor & Destructor Documentation

6.185.2.1 decaf::util::zip::CheckedInputStream::CheckedInputStream (InputStream * *inputStream*, Checksum * *sum*, bool *own* = *false*)

Create a new instance of a **CheckedInputStream** (p. 1055).

Parameters

inputStream The `InputStream` instance to Wrap.

sum The **Checksum** (p. 1059) instance to use (does not take ownership of the Pointer).

own Indicates if this filer should take ownership of the `InputStream`.

Exceptions

NullPointerException if the **Checksum** (p. 1059) pointer is NULL.

6.185.2.2 `virtual decaf::util::zip::CheckedInputStream::~~CheckedInputStream ()`
[virtual]

6.185.3 Member Function Documentation

6.185.3.1 `virtual int decaf::util::zip::CheckedInputStream::doReadArrayBounded`
(`unsigned char * buffer`, `int size`, `int offset`,
`int length`) `throw (decaf::io::IOException,`
`decaf::lang::exceptions::IndexOutOfBoundsException,`
`decaf::lang::exceptions::NullPointerException)` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p.1774).

6.185.3.2 `virtual int decaf::util::zip::CheckedInputStream::doReadByte ()` `throw`
(`decaf::io::IOException`) [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p.1774).

6.185.3.3 `Checksum* decaf::util::zip::CheckedInputStream::getChecksum ()`
`const` [inline]

Returns a Pointer to the **Checksum** (p.1059) that is in use by this **CheckedInputStream** (p.1055).

Returns

the pointer to the **Checksum** (p.1059) instance that is in use by this object.

6.185.3.4 `virtual long long decaf::util::zip::CheckedInputStream::skip (long long`
`num)` `throw (decaf::io::IOException)` [virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p.1909) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p.2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

Adds the skipped bytes into the **Checksum** (p. 1059).

Reimplemented from **decaf::io::FilterInputStream** (p. 1776).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/CheckedInputStream.h`

6.186 decaf::util::zip::CheckedOutputStream Class Reference

An implementation of a **FilterOutputStream** that will maintain a **Checksum** (p. 1059) of the bytes written, the **Checksum** (p. 1059) can then be used to verify the integrity of the output stream.

```
#include <src/main/decaf/util/zip/CheckedOutputStream.h>
```

Inheritance diagram for `decaf::util::zip::CheckedOutputStream`:

Public Member Functions

- **CheckedOutputStream** (`decaf::io::OutputStream *outputStream`, **Checksum** *sum, `bool own=false`)

*Create a new instance of a **CheckedOutputStream** (p. 1058).*

- `virtual ~CheckedOutputStream ()`
- **Checksum** * `getChecksum ()` `const`

Protected Member Functions

- `virtual void doWriteByte` (`unsigned char value`) `throw (decaf::io::IOException)`
- `virtual void doWriteArrayBounded` (`const unsigned char *buffer`, `int size`, `int offset`, `int length`) `throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

6.186.1 Detailed Description

An implementation of a **FilterOutputStream** that will maintain a **Checksum** (p. 1059) of the bytes written, the **Checksum** (p. 1059) can then be used to verify the integrity of the output stream.

Since

1.0

6.186.2 Constructor & Destructor Documentation

6.186.2.1 `decaf::util::zip::CheckedOutputStream::CheckedOutputStream (decaf::io::OutputStream * outputStream, Checksum * sum, bool own = false)`

Create a new instance of a **CheckedOutputStream** (p. 1058).

Parameters

outputStream The **OutputStream** instance to Wrap.

sum The **Checksum** (p. 1059) instance to use (does not take ownership of the Pointer).

own Indicates if this filer should take ownership of the **InputStream**.

Exceptions

NullPointerException if the **Checksum** (p. 1059) pointer is NULL.

6.186.2.2 `virtual decaf::util::zip::CheckedOutputStream::~~CheckedOutputStream () [virtual]`

6.186.3 Member Function Documentation

6.186.3.1 `virtual void decaf::util::zip::CheckedOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [protected, virtual]`

Reimplemented from **decaf::io::FilterOutputStream** (p. 1779).

6.186.3.2 `virtual void decaf::util::zip::CheckedOutputStream::doWriteByte (unsigned char value) throw (decaf::io::IOException) [protected, virtual]`

Reimplemented from **decaf::io::FilterOutputStream** (p. 1779).

6.186.3.3 `Checksum* decaf::util::zip::CheckedOutputStream::getChecksum () const [inline]`

Returns

a pointer to the **Checksum** (p. 1059) instance in use by this object.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/CheckedOutputStream.h`

6.187 decaf::util::zip::Checksum Class Reference

An interface used to represent **Checksum** (p.1059) values in the Zip package.

```
#include <src/main/decaf/util/zip/Checksum.h>
```

Inheritance diagram for decaf::util::zip::Checksum:

Public Member Functions

- virtual **~Checksum** ()
- virtual long long **getValue** () const =0
- virtual void **reset** ()=0
Reset the checksum to its initial value.
- virtual void **update** (const std::vector< unsigned char > &buffer)=0
Updates the current checksum with the specified vector of bytes.
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length)=0
 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Updates the current checksum with the specified array of bytes.
- virtual void **update** (const unsigned char *buffer, int size, int offset, int length)=0 throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Updates the current checksum with the specified array of bytes.
- virtual void **update** (int byte)=0
Updates the current checksum with the specified byte value.

6.187.1 Detailed Description

An interface used to represent **Checksum** (p.1059) values in the Zip package.

Since

1.0

6.187.2 Constructor & Destructor Documentation

6.187.2.1 virtual decaf::util::zip::Checksum::~~Checksum () [inline, virtual]

6.187.3 Member Function Documentation

6.187.3.1 virtual long long decaf::util::zip::Checksum::getValue () const [pure virtual]

Returns

the current checksum value.

Implemented in `decaf::util::zip::Adler32` (p. 664), and `decaf::util::zip::CRC32` (p. 1421).

6.187.3.2 virtual void decaf::util::zip::Checksum::reset () [pure virtual]

Reset the checksum to its initial value.

Implemented in `decaf::util::zip::Adler32` (p. 664), and `decaf::util::zip::CRC32` (p. 1421).

6.187.3.3 virtual void decaf::util::zip::Checksum::update (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer The buffer to read the updated bytes from.

size The size of the passed buffer.

offset The position in the buffer to start reading.

length The amount of data to read from the byte buffer.

Exceptions

NullPointerException if the passed buffer is NULL.

IndexOutOfBoundsException if `offset + length > size` of the buffer.

Implemented in `decaf::util::zip::Adler32` (p. 665), and `decaf::util::zip::CRC32` (p. 1421).

6.187.3.4 virtual void decaf::util::zip::Checksum::update (const std::vector< unsigned char > & *buffer*, int *offset*, int *length*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer The buffer to read the updated bytes from.

offset The position in the buffer to start reading.

length The amount of data to read from the byte buffer.

Exceptions

IndexOutOfBoundsException if `offset + length > size` of the buffer.

Implemented in `decaf::util::zip::Adler32` (p. 665), and `decaf::util::zip::CRC32` (p. 1421).

6.187.3.5 `virtual void decaf::util::zip::Checksum::update (const std::vector< unsigned char > & buffer)` [pure virtual]

Updates the current checksum with the specified vector of bytes.

Parameters

buffer The buffer to read the updated bytes from.

Implemented in `decaf::util::zip::Adler32` (p. 665), and `decaf::util::zip::CRC32` (p. 1422).

6.187.3.6 `virtual void decaf::util::zip::Checksum::update (int byte)` [pure virtual]

Updates the current checksum with the specified byte value.

Parameters

byte The byte value to update the current `Checksum` (p. 1059) with (0..255).

Implemented in `decaf::util::zip::Adler32` (p. 666), and `decaf::util::zip::CRC32` (p. 1422).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/Checksum.h`

6.188 decaf::lang::exceptions::ClassCastException Class Reference

```
#include <src/main/decaf/lang/exceptions/ClassCastException.h>
```

Inheritance diagram for `decaf::lang::exceptions::ClassCastException`:

Public Member Functions

- `ClassCastException ()` throw ()
Default Constructor.
- `ClassCastException (const Exception &ex)` throw ()
*Conversion Constructor from some other **Exception** (p. 1712).*
- `ClassCastException (const ClassCastException &ex)` throw ()
Copy Constructor.
- `ClassCastException (const std::exception *cause)` throw ()
Constructor.
- `ClassCastException (const char *file, const int lineNumber, const char *msg,...)` throw ()

Constructor - Initializes the file name and line number where this message occurred.

- **ClassCastException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **ClassCastException** * clone () const

Clones this exception.

- virtual ~**ClassCastException** () throw ()

6.188.1 Constructor & Destructor Documentation

6.188.1.1 decaf::lang::exceptions::ClassCastException::ClassCastException () throw () [inline]

Default Constructor.

6.188.1.2 decaf::lang::exceptions::ClassCastException::ClassCastException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.188.1.3 decaf::lang::exceptions::ClassCastException::ClassCastException (const ClassCastException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.188.1.4 decaf::lang::exceptions::ClassCastException::ClassCastException (const std::exception * cause) throw () [inline]

Constructor.

Parameters

cause **Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.188.1.5 `decaf::lang::exceptions::ClassCastException::ClassCastException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.188.1.6 `decaf::lang::exceptions::ClassCastException::ClassCastException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.188.1.7 `virtual decaf::lang::exceptions::ClassCastException::~~ClassCastException () throw () [inline, virtual]`

6.188.2 Member Function Documentation

6.188.2.1 `virtual ClassCastException* decaf::lang::exceptions::ClassCastException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/ClassCastException.h`

6.189 cms::Closeable Class Reference

Interface for a class that implements the close method.

```
#include <src/main/cms/Closeable.h>
```

Inheritance diagram for cms::Closeable:

Public Member Functions

- virtual `~Closeable()`
- virtual void `close()` throw (CMSEException)
Closes this object and deallocates the appropriate resources.

6.189.1 Detailed Description

Interface for a class that implements the close method. A class that implements this interface should release all resources upon the close call and should throw an exception from any methods that require those resources after they have been closed.

Since

1.0

6.189.2 Constructor & Destructor Documentation

6.189.2.1 virtual cms::Closeable::~~Closeable () [inline, virtual]

6.189.3 Member Function Documentation

6.189.3.1 virtual void cms::Closeable::close () throw (CMSEException) [pure virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

CMSEException (p. 1074) - If an error occurs while the resource is being closed.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 994), `activemq::cmsutil::CachedProducer` (p. 998), `activemq::cmsutil::PooledSession` (p. 2768), `activemq::commands::ActiveMQTempDestination` (p. 525), `activemq::core::ActiveMQProducer` (p. 425), `cms::Connection` (p. 1169), and `cms::Session` (p. 3152).

The documentation for this class was generated from the following file:

- `src/main/cms/Closeable.h`

6.190 decaf::io::Closeable Class Reference

Interface for a class that implements the close method.

```
#include <src/main/decaf/io/Closeable.h>
```

Inheritance diagram for decaf::io::Closeable:

Public Member Functions

- virtual `~Closeable()`
- virtual void `close()` = 0 throw (io::IOException)
Closes this object and deallocates the appropriate resources.

6.190.1 Detailed Description

Interface for a class that implements the close method.

6.190.2 Constructor & Destructor Documentation

6.190.2.1 virtual decaf::io::Closeable::~~Closeable () [inline, virtual]

6.190.3 Member Function Documentation

6.190.3.1 virtual void decaf::io::Closeable::close () throw (io::IOException)
[pure virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException (p. 2003) if an error occurs while closing.

Implemented in `activemq::transport::mock::MockTransport` (p. 2594), `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2679), `decaf::net::Socket` (p. 3287), `decaf::util::logging::ConsoleHandler` (p. 1301), and `decaf::util::logging::StreamHandler` (p. 3414).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Closeable.h`

6.191 activemq::transport::failover::CloseTransportsTask Class Reference

```
#include <src/main/activemq/transport/failover/CloseTransportsTask.h>
```

Inheritance diagram for `activemq::transport::failover::CloseTransportsTask`:

Public Member Functions

- **CloseTransportsTask** ()
- virtual **~CloseTransportsTask** ()
- void **add** (const **Pointer**< **Transport** > &transport)
*Add a new **Transport** (p. 3629) to close.*
- virtual bool **isPending** () const
This Task is pending if there are transports in the Queue that need to be closed.
- virtual bool **iterate** ()
Return true until all transports have been closed and removed from the queue.

6.191.1 Constructor & Destructor Documentation

6.191.1.1 `activemq::transport::failover::CloseTransportsTask::CloseTransportsTask ()`

6.191.1.2 `virtual
activemq::transport::failover::CloseTransportsTask::~~CloseTransportsTask
() [virtual]`

6.191.2 Member Function Documentation

6.191.2.1 `void activemq::transport::failover::CloseTransportsTask::add (const
Pointer< Transport > & transport)`

Add a new **Transport** (p. 3629) to close.

6.191.2.2 `virtual bool activemq::transport::failover::CloseTransportsTask::isPending
() const [virtual]`

This Task is pending if there are transports in the Queue that need to be closed.

Returns

true if there is a transport in the queue that needs closed.

Implements `activemq::threads::CompositeTask` (p. 1132).

6.191.2.3 `virtual bool activemq::transport::failover::CloseTransportsTask::iterate () [virtual]`

Return true until all transports have been closed and removed from the queue.

Implements `activemq::threads::Task` (p. 3495).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/CloseTransportsTask.h`

6.192 activemq::cmsutil::CmsAccessor Class Reference

Base class for `activemq.cmsutil.CmsTemplate` (p.1083) and other CMS-accessing gateway helpers, defining common properties such as the CMS `cms.ConnectionFactory` (p.1228) to operate on.

```
#include <src/main/activemq/cmsutil/CmsAccessor.h>
```

Inheritance diagram for `activemq::cmsutil::CmsAccessor`:

Public Member Functions

- **CmsAccessor** ()
- virtual **~CmsAccessor** ()
- virtual **ResourceLifecycleManager** * **getResourceLifecycleManager** ()
- virtual const **ResourceLifecycleManager** * **getResourceLifecycleManager** () const
- virtual void **setConnectionFactory** (**cms::ConnectionFactory** *connectionFactory)
Set the ConnectionFactory to use for obtaining CMS Connections.
- virtual const **cms::ConnectionFactory** * **getConnectionFactory** () const
Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.
- virtual **cms::ConnectionFactory** * **getConnectionFactory** ()
Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.
- virtual void **setSessionAcknowledgeMode** (**cms::Session::AcknowledgeMode** sessionAcknowledgeMode)
Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.
- virtual **cms::Session::AcknowledgeMode** **getSessionAcknowledgeMode** () const
Return the acknowledgment mode for CMS sessions.

Protected Member Functions

- **CmsAccessor** (const **CmsAccessor** &)
- **CmsAccessor** & **operator=** (const **CmsAccessor** &)
- virtual void **init** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)
Initializes this object and prepares it for use.
- virtual void **destroy** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)
Shuts down this object and destroys any allocated resources.

- virtual **cms::Connection * createConnection** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

Create a CMS Connection via this template's ConnectionFactory.

- virtual **cms::Session * createSession** (cms::Connection *con) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

Create a CMS Session for the given Connection.

- virtual void **checkConnectionFactory** () throw (decaf::lang::exceptions::IllegalStateException)

Verifies that the connection factory is valid.

6.192.1 Detailed Description

Base class for **activemq.cmsutil.CmsTemplate** (p.1083) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p.1228) to operate on. The subclass **activemq.cmsutil.CmsDestinationAccessor** (p.1071) adds further, destination-related properties.

Not intended to be used directly.

See also

activemq.cmsutil.CmsDestinationAccessor (p.1071)

activemq.cmsutil.CmsTemplate (p.1083)

6.192.2 Constructor & Destructor Documentation

6.192.2.1 **activemq::cmsutil::CmsAccessor::CmsAccessor** (const CmsAccessor &) [inline, protected]

6.192.2.2 **activemq::cmsutil::CmsAccessor::CmsAccessor** ()

6.192.2.3 **virtual activemq::cmsutil::CmsAccessor::~~CmsAccessor** () [virtual]

6.192.3 Member Function Documentation

6.192.3.1 **virtual void activemq::cmsutil::CmsAccessor::checkConnectionFactory** () throw (decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Verifies that the connection factory is valid.

6.192.3.2 **virtual cms::Connection* activemq::cmsutil::CmsAccessor::createConnection** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Create a CMS Connection via this template's ConnectionFactory.

Returns

the new CMS Connection

Exceptions

cms::CMSException (p. 1074) if thrown by CMS API methods

6.192.3.3 `virtual cms::Session* activemq::cmsutil::CmsAccessor::createSession (cms::Connection * con) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]`

Create a CMS Session for the given Connection.

Parameters

con the CMS Connection to create a Session for

Returns

the new CMS Session

Exceptions

cms::CMSException (p. 1074) if thrown by CMS API methods

6.192.3.4 `virtual void activemq::cmsutil::CmsAccessor::destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [inline, protected, virtual]`

Shuts down this object and destroys any allocated resources.

Reimplemented in `activemq::cmsutil::CmsDestinationAccessor` (p.1073), and `activemq::cmsutil::CmsTemplate` (p.1087).

6.192.3.5 `virtual const cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () const [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.192.3.6 `virtual cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

- 6.192.3.7** `virtual ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager ()`
[inline, virtual]
- 6.192.3.8** `virtual const ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager () const`
[inline, virtual]
- 6.192.3.9** `virtual cms::Session::AcknowledgeMode activemq::cmsutil::CmsAccessor::getSessionAcknowledgeMode () const`
[inline, virtual]

Return the acknowledgment mode for CMS sessions.

Returns

the acknowledgment mode applied by this accessor

- 6.192.3.10** `virtual void activemq::cmsutil::CmsAccessor::init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)`
[inline, protected, virtual]

Initializes this object and prepares it for use.

This should be called before any other methods are called. This version does nothing.

Reimplemented in `activemq::cmsutil::CmsDestinationAccessor` (p.1073), and `activemq::cmsutil::CmsTemplate` (p.1090).

- 6.192.3.11** `CmsAccessor& activemq::cmsutil::CmsAccessor::operator= (const CmsAccessor &)` [inline, protected]
- 6.192.3.12** `virtual void activemq::cmsutil::CmsAccessor::setConnectionFactory (cms::ConnectionFactory * connectionFactory)` [inline, virtual]

Set the ConnectionFactory to use for obtaining CMS Connections.

- 6.192.3.13** `virtual void activemq::cmsutil::CmsAccessor::setSessionAcknowledgeMode (cms::Session::AcknowledgeMode sessionAcknowledgeMode)` [inline, virtual]

Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message. Default is `AUTO_ACKNOWLEDGE`.

Parameters

sessionAcknowledgeMode the acknowledgment mode

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsAccessor.h`

6.193 activemq::cmsutil::CmsDestinationAccessor Class Reference

Extends the `CmsAccessor` (p. 1068) to add support for resolving destination names.

```
#include <src/main/activemq/cmsutil/CmsDestinationAccessor.h>
```

Inheritance diagram for `activemq::cmsutil::CmsDestinationAccessor`:

Public Member Functions

- **CmsDestinationAccessor** ()
- virtual **~CmsDestinationAccessor** ()
- virtual bool **isPubSubDomain** () const
- virtual void **setPubSubDomain** (bool pubSubDomain)
- virtual **DestinationResolver** * **getDestinationResolver** ()
- virtual const **DestinationResolver** * **getDestinationResolver** () const
- virtual void **setDestinationResolver** (**DestinationResolver** *destRes)

Protected Member Functions

- **CmsDestinationAccessor** (const **CmsDestinationAccessor** &)
- **CmsDestinationAccessor** & **operator=** (const **CmsDestinationAccessor** &)
- virtual void **init** () throw (cms::CMSEException, decaf::lang::exceptions::IllegalStateException)
Initializes the destination resolver.
- virtual void **destroy** () throw (cms::CMSEException, decaf::lang::exceptions::IllegalStateException)
*Calls **destroy()** (p. 1073) on the destination resolver.*
- virtual **cms::Destination** * **resolveDestinationName** (**cms::Session** *session, const std::string &destName) throw (cms::CMSEException, decaf::lang::exceptions::IllegalStateException)
*Resolves the destination via the *DestinationResolver* (p. 1642).*
- virtual void **checkDestinationResolver** () throw (decaf::lang::exceptions::IllegalStateException)
Verifies that the destination resolver is valid.

6.193.1 Detailed Description

Extends the `CmsAccessor` (p. 1068) to add support for resolving destination names. Not intended to be used directly.

See also

CmsTemplate (p. 1083)
CmsAccessor (p. 1068)

6.193.2 Constructor & Destructor Documentation

6.193.2.1 `activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor (const CmsDestinationAccessor &)` [inline, protected]

6.193.2.2 `activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor ()`

6.193.2.3 `virtual
activemq::cmsutil::CmsDestinationAccessor::~~CmsDestinationAccessor ()` [virtual]

6.193.3 Member Function Documentation

6.193.3.1 `virtual void activemq::cmsutil::CmsDestinationAccessor::checkDestinationResolver () throw (decaf::lang::exceptions::IllegalStateException)` [protected, virtual]

Verifies that the destination resolver is valid.

6.193.3.2 `virtual void activemq::cmsutil::CmsDestinationAccessor::destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)` [protected, virtual]

Calls `destroy()` (p. 1073) on the destination resolver.

Reimplemented from `activemq::cmsutil::CmsAccessor` (p. 1070).

Reimplemented in `activemq::cmsutil::CmsTemplate` (p. 1087).

6.193.3.3 `virtual const DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver () const` [inline, virtual]

6.193.3.4 `virtual DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ()` [inline, virtual]

6.193.3.5 `virtual void activemq::cmsutil::CmsDestinationAccessor::init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)` [protected, virtual]

Initializes the destination resolver.

Reimplemented from `activemq::cmsutil::CmsAccessor` (p. 1071).

Reimplemented in `activemq::cmsutil::CmsTemplate` (p. 1090).

6.193.3.6 virtual bool activemq::cmsutil::CmsDestinationAccessor::isPubSubDomain () const
[inline, virtual]

6.193.3.7 CmsDestinationAccessor& activemq::cmsutil::CmsDestinationAccessor::operator= (const CmsDestinationAccessor &) [inline, protected]

6.193.3.8 virtual cms::Destination* activemq::cmsutil::CmsDestinationAccessor::resolveDestinationName (cms::Session * *session*, const std::string & *destName*) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Resolves the destination via the DestinationResolver (p. 1642).

Parameters

session the session
destName the name of the destination.

Returns

the destination

Exceptions

cms::CMSException (p. 1074) if resolution failed.
decaf::lang::exceptions::IllegalStateException (p. 1869) if the destination resolver property is NULL.

6.193.3.9 virtual void activemq::cmsutil::CmsDestinationAccessor::setDestinationResolver (DestinationResolver * *destRes*) [inline, virtual]

6.193.3.10 virtual void activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain (bool *pubSubDomain*) [inline, virtual]

Reimplemented in `activemq::cmsutil::CmsTemplate` (p. 1095).

Referenced by `activemq::cmsutil::CmsTemplate::setPubSubDomain()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsDestinationAccessor.h`

6.194 cms::CMSException Class Reference

CMS API Exception that is the base for all exceptions thrown from CMS classes.

`#include <src/main/cms/CMSException.h>`

Inheritance diagram for `cms::CMSException`:

Public Member Functions

- **CMSException** () throw ()
- **CMSException** (const **CMSException** &ex) throw ()
- **CMSException** (const std::string &message, const std::exception *cause) throw ()
- **CMSException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSException** () throw ()
- virtual std::string **getMessage** () const
Gets the cause of the error.
- virtual const std::exception * **getCause** () const
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const
Provides the stack trace for every point where this exception was caught, marked, and rethrown.
- virtual void **setMark** (const char *file, const int lineNumber)
Adds a file/line number to the stack trace.
- virtual void **printStackTrace** () const
Prints the stack trace to std::err.
- virtual void **printStackTrace** (std::ostream &stream) const
Prints the stack trace to the given output stream.
- virtual std::string **getStackTraceString** () const
Gets the stack trace as one contiguous string.
- virtual const char * **what** () const throw ()
*Overloads the std::exception **what**() (p. 1077) function to return the cause of the exception.*

6.194.1 Detailed Description

CMS API Exception that is the base for all exceptions thrown from CMS classes. This class represents an error that has occurred in CMS, providers can wrap provider specific exceptions in this class by setting the cause to an instance of a provider specific exception provided it can be cast to an std::exception.

Since the contained cause exception is of type std::exception and the C++ exception class has no clone or copy method defined the contained exception can only be owned by one instance of an **CMSException** (p. 1074). To that end the class hands off the exception to each successive copy so care must be taken when handling **CMSException** (p. 1074) instances.

Since

1.0

6.194.2 Constructor & Destructor Documentation

- 6.194.2.1** cms::CMSException::CMSException () throw ()
- 6.194.2.2** cms::CMSException::CMSException (const CMSException & *ex*) throw ()
- 6.194.2.3** cms::CMSException::CMSException (const std::string & *message*, const std::exception * *cause*) throw ()
- 6.194.2.4** cms::CMSException::CMSException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.194.2.5** virtual cms::CMSException::~~CMSException () throw () [virtual]

6.194.3 Member Function Documentation

- 6.194.3.1** virtual const std::exception* cms::CMSException::getCause () const [virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

- 6.194.3.2** virtual std::string cms::CMSException::getMessage () const [virtual]

Gets the cause of the error.

Returns

string errors message

- 6.194.3.3** virtual std::vector< std::pair< std::string, int> > cms::CMSException::getStackTrace () const [virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

Returns

vector containing stack trace strings

6.194.3.4 `virtual std::string cms::CMSException::getStackTraceString () const`
[virtual]

Gets the stack trace as one contiguous string.

Returns

string with formatted stack trace data

6.194.3.5 `virtual void cms::CMSException::printStackTrace () const` [virtual]

Prints the stack trace to std::err.

6.194.3.6 `virtual void cms::CMSException::printStackTrace (std::ostream &
stream) const` [virtual]

Prints the stack trace to the given output stream.

Parameters

stream the target output stream.

6.194.3.7 `virtual void cms::CMSException::setMark (const char * file, const int
lineNumber)` [virtual]

Adds a file/line number to the stack trace.

Parameters

file The name of the file calling this method (use `__FILE__`).

lineNumber The line number in the calling file (use `__LINE__`).

6.194.3.8 `virtual const char* cms::CMSException::what () const throw ()`
[virtual]

Overloads the std::exception `what()` (p. 1077) function to return the cause of the exception.

Returns

const char pointer to error message

The documentation for this class was generated from the following file:

- `src/main/cms/CMSException.h`

6.195 `activemq::util::CMSExceptionSupport` Class Reference

```
#include <src/main/activemq/util/CMSExceptionSupport.h>
```

Public Member Functions

- virtual `~CMSExceptionSupport ()`

Static Public Member Functions

- static `cms::CMSException create (const std::string &msg, const decaf::lang::Exception &cause)`
- static `cms::CMSException create (const decaf::lang::Exception &cause)`
- static `cms::MessageEOFException createMessageEOFException (const decaf::lang::Exception &cause)`
- static `cms::MessageFormatException createMessageFormatException (const decaf::lang::Exception &cause)`

6.195.1 Constructor & Destructor Documentation

- 6.195.1.1** virtual `activemq::util::CMSExceptionSupport::~~CMSExceptionSupport ()` [virtual]

6.195.2 Member Function Documentation

- 6.195.2.1** static `cms::CMSException activemq::util::CMSExceptionSupport::create (const std::string & msg, const decaf::lang::Exception & cause)` [static]
- 6.195.2.2** static `cms::CMSException activemq::util::CMSExceptionSupport::create (const decaf::lang::Exception & cause)` [static]
- 6.195.2.3** static `cms::MessageEOFException activemq::util::CMSExceptionSupport::createMessageEOFException (const decaf::lang::Exception & cause)` [static]
- 6.195.2.4** static `cms::MessageFormatException activemq::util::CMSExceptionSupport::createMessageFormatException (const decaf::lang::Exception & cause)` [static]

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBooleanProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getByteProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getDoubleProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getFloatProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getIntProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getLongProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getShortProperty()`, and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getStringProperty()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CMSExceptionSupport.h`

6.196 cms::CMSProperties Class Reference

Interface for a Java-like properties object.

```
#include <src/main/cms/CMSProperties.h>
```

Inheritance diagram for cms::CMSProperties:

Public Member Functions

- virtual `~CMSProperties ()`
- virtual bool `isEmpty () const =0`
Returns true if the properties object is empty.
- virtual const char * `getProperty (const std::string &name) const =0`
Looks up the value for the given property.
- virtual std::string `getProperty (const std::string &name, const std::string &defaultValue) const =0`
Looks up the value for the given property.
- virtual void `setProperty (const std::string &name, const std::string &value)=0`
Sets the value for a given property.
- virtual bool `hasProperty (const std::string &name) const =0`
Check to see if the Property exists in the set.
- virtual void `remove (const std::string &name)=0`
Removes the property with the given name.
- virtual std::vector< std::pair< std::string, std::string > > `toArray () const =0`
Method that serializes the contents of the property map to an array.
- virtual void `copy (const CMSProperties *source)=0`
Copies the contents of the given properties object to this one.
- virtual `CMSProperties * clone () const =0`
Clones this object.
- virtual void `clear ()=0`
Clears all properties from the map.
- virtual std::string `toString () const =0`
Formats the contents of the Properties Object into a string that can be logged, etc.

6.196.1 Detailed Description

Interface for a Java-like properties object. This is essentially a map of key-value string pairs.

Since

1.1

6.196.2 Constructor & Destructor Documentation

6.196.2.1 virtual cms::CMSProperties::~~CMSProperties () [inline, virtual]

6.196.3 Member Function Documentation

6.196.3.1 virtual void cms::CMSProperties::clear () [pure virtual]

Clears all properties from the map.

Implemented in **activemq::util::ActiveMQProperties** (p. 432).

6.196.3.2 virtual CMSProperties* cms::CMSProperties::clone () const [pure virtual]

Clones this object.

Returns

a replica of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 432).

6.196.3.3 virtual void cms::CMSProperties::copy (const CMSProperties * *source*) [pure virtual]

Copies the contents of the given properties object to this one.

Parameters

source The source properties object.

6.196.3.4 virtual const char* cms::CMSProperties::getProperty (const std::string & *name*) const [pure virtual]

Looks up the value for the given property.

Parameters

name The name of the property to be looked up.

Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implemented in **activemq::util::ActiveMQProperties** (p. 433).

6.196.3.5 `virtual std::string cms::CMSProperties::getProperty (const std::string & name, const std::string & defaultValue) const` [pure virtual]

Looks up the value for the given property.

Parameters

name the name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implemented in `activemq::util::ActiveMQProperties` (p. 433).

6.196.3.6 `virtual bool cms::CMSProperties::hasProperty (const std::string & name) const` [pure virtual]

Check to see if the Property exists in the set.

Parameters

name the name of the property to check

Returns

true if property exists, false otherwise.

Implemented in `activemq::util::ActiveMQProperties` (p. 433).

6.196.3.7 `virtual bool cms::CMSProperties::isEmpty () const` [pure virtual]

Returns true if the properties object is empty.

Returns

true if empty

Implemented in `activemq::util::ActiveMQProperties` (p. 434).

6.196.3.8 `virtual void cms::CMSProperties::remove (const std::string & name)` [pure virtual]

Removes the property with the given name.

Parameters

name the name of the property to be removed.s

Implemented in `activemq::util::ActiveMQProperties` (p. 434).

6.196.3.9 `virtual void cms::CMSProperties::setProperty (const std::string & name, const std::string & value) [pure virtual]`

Sets the value for a given property.

If the property already exists, overwrites the value.

Parameters

name The name of the value to be written.

value The value to be written.

Implemented in `activemq::util::ActiveMQProperties` (p. 434).

6.196.3.10 `virtual std::vector< std::pair< std::string, std::string > > cms::CMSProperties::toArray () const [pure virtual]`

Method that serializes the contents of the property map to an array.

Returns

list of pairs where the first is the name and the second is the value.

Implemented in `activemq::util::ActiveMQProperties` (p. 434).

6.196.3.11 `virtual std::string cms::CMSProperties::toString () const [pure virtual]`

Formats the contents of the Properties Object into a string that can be logged, etc.

Returns

string value of this object.

Implemented in `activemq::util::ActiveMQProperties` (p. 435).

The documentation for this class was generated from the following file:

- `src/main/cms/CMSProperties.h`

6.197 cms::CMSSecurityException Class Reference

This exception must be thrown when a provider rejects a user name/password submitted by a client.

```
#include <src/main/cms/CMSSecurityException.h>
```

Inheritance diagram for cms::CMSSecurityException:

Public Member Functions

- **CMSSecurityException** () throw ()
- **CMSSecurityException** (const **CMSSecurityException** &ex) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception *cause) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSSecurityException** () throw ()

6.197.1 Detailed Description

This exception must be thrown when a provider rejects a user name/password submitted by a client. It may also be thrown for any case where a security restriction prevents a method from completing.

Since

1.3

6.197.2 Constructor & Destructor Documentation

- 6.197.2.1** `cms::CMSSecurityException::CMSSecurityException () throw ()`
- 6.197.2.2** `cms::CMSSecurityException::CMSSecurityException (const CMSSecurityException & ex) throw ()`
- 6.197.2.3** `cms::CMSSecurityException::CMSSecurityException (const std::string & message, const std::exception * cause) throw ()`
- 6.197.2.4** `cms::CMSSecurityException::CMSSecurityException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.197.2.5** `virtual cms::CMSSecurityException::~~CMSSecurityException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/CMSSecurityException.h`

6.198 activemq::cmsutil::CmsTemplate Class Reference

CmsTemplate (p. 1083) simplifies performing synchronous CMS operations.

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate`:

Data Structures

- class **ProducerExecutor**
- class **ReceiveExecutor**
- class **ResolveProducerExecutor**
- class **ResolveReceiveExecutor**
- class **SendExecutor**

Public Member Functions

- **CmsTemplate** ()
- **CmsTemplate** (cms::ConnectionFactory *connectionFactory)
- virtual **~CmsTemplate** ()
- virtual void **setDefaultDestination** (cms::Destination *defaultDestination)
Sets the destination object to be used by default for send/receive operations.
- virtual const cms::Destination * **getDefaultDestination** () const
Retrieves the default destination to be used for send/receive operations.
- virtual cms::Destination * **getDefaultDestination** ()
Retrieves the default destination to be used for send/receive operations.
- virtual void **setDefaultDestinationName** (const std::string &defaultDestinationName)
Sets the name of the default destination to be used from send/receive operations.
- virtual const std::string **getDefaultDestinationName** () const
Gets the name of the default destination to be used for send/receive operations.
- virtual void **setPubSubDomain** (bool pubSubDomain)
Indicates whether the default destination is a topic (true) or a queue (false).
- virtual void **setMessageIdEnabled** (bool messageIdEnabled)
- virtual bool **isMessageIdEnabled** () const
- virtual void **setMessageTimestampEnabled** (bool messageTimestampEnabled)
- virtual bool **isMessageTimestampEnabled** () const
- virtual void **setNoLocal** (bool noLocal)
- virtual bool **isNoLocal** () const
- virtual void **setReceiveTimeout** (long long receiveTimeout)
- virtual long long **getReceiveTimeout** () const
- virtual void **setExplicitQosEnabled** (bool explicitQosEnabled)
Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.
- virtual bool **isExplicitQosEnabled** () const
If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message.
- virtual void **setDeliveryPersistent** (bool deliveryPersistent)
Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").

- virtual void **setDeliveryMode** (int deliveryMode)
Set the delivery mode to use when sending a message.
- virtual int **getDeliveryMode** () const
Return the delivery mode to use when sending a message.
- virtual void **setPriority** (int priority)
Set the priority of a message when sending.
- virtual int **getPriority** () const
Return the priority of a message when sending.
- virtual void **setTimeToLive** (long long timeToLive)
Set the time-to-live of the message when sending.
- virtual long long **getTimeToLive** () const
Return the time-to-live of the message when sending.
- virtual void **execute** (**SessionCallback** *action) throw (cms::CMSEException)
Executes the given action within a CMS Session.
- virtual void **execute** (**ProducerCallback** *action) throw (cms::CMSEException)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **execute** (cms::Destination *dest, **ProducerCallback** *action) throw (cms::CMSEException)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **execute** (const std::string &destinationName, **ProducerCallback** *action) throw (cms::CMSEException)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **send** (**MessageCreator** *messageCreator) throw (cms::CMSEException)
Convenience method for sending a message to the default destination.
- virtual void **send** (cms::Destination *dest, **MessageCreator** *messageCreator) throw (cms::CMSEException)
Convenience method for sending a message to the specified destination.
- virtual void **send** (const std::string &destinationName, **MessageCreator** *messageCreator) throw (cms::CMSEException)
Convenience method for sending a message to the specified destination.
- virtual cms::Message * **receive** () throw (cms::CMSEException)
Performs a synchronous read from the default destination.
- virtual cms::Message * **receive** (cms::Destination *destination) throw (cms::CMSEException)
Performs a synchronous read from the specified destination.

- virtual **cms::Message** * **receive** (const std::string &destinationName) throw (cms::CMSEException)
Performs a synchronous read from the specified destination.
- virtual **cms::Message** * **receiveSelected** (const std::string &selector) throw (cms::CMSEException)
Performs a synchronous read consuming only messages identified by the given selector.
- virtual **cms::Message** * **receiveSelected** (cms::Destination *destination, const std::string &selector) throw (cms::CMSEException)
Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.
- virtual **cms::Message** * **receiveSelected** (const std::string &destinationName, const std::string &selector) throw (cms::CMSEException)
Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Static Public Attributes

- static const long long **RECEIVE_TIMEOUT_NO_WAIT**
Timeout value indicating that a receive operation should check if a message is immediately available without blocking.
- static const long long **RECEIVE_TIMEOUT_INDEFINITE_WAIT**
Timeout value indicating a blocking receive without timeout.
- static const int **DEFAULT_PRIORITY**
Default message priority.
- static const long long **DEFAULT_TIME_TO_LIVE**
My default, messages should live forever.

Protected Member Functions

- **CmsTemplate** (const **CmsTemplate** &)
- **CmsTemplate** & **operator=** (const **CmsTemplate** &)
- void **init** () throw (cms::CMSEException, decaf::lang::exceptions::IllegalStateException)
Initializes this object and prepares it for use.
- void **destroy** () throw (cms::CMSEException, decaf::lang::exceptions::IllegalStateException)
Clears all internal resources.

Friends

- class **ProducerExecutor**
- class **ResolveProducerExecutor**
- class **SendExecutor**
- class **ReceiveExecutor**
- class **ResolveReceiveExecutor**

6.198.1 Detailed Description

CmsTemplate (p. 1083) simplifies performing synchronous CMS operations. This class is intended to be for CMS what Spring's **JmsTemplate** is for JMS. Provided with a CMS **ConnectionFactory**, creates and manages all other CMS resources internally.

Before using **CmsTemplate** (p. 1083) the user must first set the destination (either by name or by setting the destination object directly) and then call **init** to initialize the object for use.

CmsTemplate (p.1083) allows the user to get access to a CMS **Session** through a user-defined **SessionCallback** (p.3160). Similarly, if the user wants direct access to a CMS **MessageProducer**, it can provide a **ProducerCallback** (p.2866). As a convenience, the user can bypass having to provide callbacks altogether for sending messages, by calling one of the **send** methods.

See also

SessionCallback (p. 3160)
ProducerCallback (p. 2866)
MessageCreator (p. 2426)

6.198.2 Constructor & Destructor Documentation

6.198.2.1 `activemq::cmsutil::CmsTemplate::CmsTemplate (const CmsTemplate &) [inline, protected]`

6.198.2.2 `activemq::cmsutil::CmsTemplate::CmsTemplate ()`

6.198.2.3 `activemq::cmsutil::CmsTemplate::CmsTemplate (cms::ConnectionFactory * connectionFactory)`

6.198.2.4 `virtual activemq::cmsutil::CmsTemplate::~~CmsTemplate () [virtual]`

6.198.3 Member Function Documentation

6.198.3.1 `void activemq::cmsutil::CmsTemplate::destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]`

Clears all internal resources.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 1073).

6.198.3.2 virtual void activemq::cmsutil::CmsTemplate::execute (SessionCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action within a CMS Session.

Parameters

action the action to perform within a CMS Session

Exceptions

cms::CMSException (p. 1074) thrown if an error occurs.

6.198.3.3 virtual void activemq::cmsutil::CmsTemplate::execute (ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters

action the action to perform

Exceptions

cms::CMSException (p. 1074) thrown if an error occurs.

6.198.3.4 virtual void activemq::cmsutil::CmsTemplate::execute (cms::Destination * *dest*, ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters

dest the destination to send messages to

action the action to perform

Exceptions

cms::CMSException (p. 1074) thrown if an error occurs.

6.198.3.5 virtual void activemq::cmsutil::CmsTemplate::execute (const std::string & *destinationName*, ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters

destinationName the name of the destination to send messages to (to internally be resolved to an actual destination)

action the action to perform

Exceptions

cms::CMSException (p. 1074) thrown if an error occurs.

6.198.3.6 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination () [inline, virtual]`

Retrieves the default destination to be used for send/receive operations.

Returns

the default destination. Non-const version of this method.

6.198.3.7 `virtual const cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination () const [inline, virtual]`

Retrieves the default destination to be used for send/receive operations.

Returns

the default destination. Const version of this method.

6.198.3.8 `virtual const std::string activemq::cmsutil::CmsTemplate::getDefaultDestinationName () const [inline, virtual]`

Gets the name of the default destination to be used for send/receive operations.

The destination type (topic/queue) is determined by the `pubSubDomain` property.

Returns

the default name of the destination for send/receive operations.

6.198.3.9 `virtual int activemq::cmsutil::CmsTemplate::getDeliveryMode () const [inline, virtual]`

Return the delivery mode to use when sending a message.

6.198.3.10 `virtual int activemq::cmsutil::CmsTemplate::getPriority () const [inline, virtual]`

Return the priority of a message when sending.

6.198.3.11 `virtual long long activemq::cmsutil::CmsTemplate::getReceiveTimeout () const [inline, virtual]`

6.198.3.12 `virtual long long activemq::cmsutil::CmsTemplate::getTimeToLive () const [inline, virtual]`

Return the time-to-live of the message when sending.

6.198.3.13 `void activemq::cmsutil::CmsTemplate::init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]`

Initializes this object and prepares it for use.

This should be called before any other methods are called.

Reimplemented from `activemq::cmsutil::CmsDestinationAccessor` (p. 1073).

6.198.3.14 `virtual bool activemq::cmsutil::CmsTemplate::isExplicitQosEnabled () const [inline, virtual]`

If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message.

Otherwise, the default values, that may be set administratively, will be used.

Returns

true if overriding default values of QOS parameters (deliveryMode, priority, and timeToLive)

See also

`setDeliveryMode` (p. 1094)

`setPriority` (p. 1095)

`setTimeToLive` (p. 1095)

6.198.3.15 `virtual bool activemq::cmsutil::CmsTemplate::isMessageIdEnabled () const [inline, virtual]`

6.198.3.16 `virtual bool activemq::cmsutil::CmsTemplate::isMessageTimestampEnabled () const [inline, virtual]`

6.198.3.17 `virtual bool activemq::cmsutil::CmsTemplate::isNoLocal () const [inline, virtual]`

6.198.3.18 `CmsTemplate& activemq::cmsutil::CmsTemplate::operator= (const CmsTemplate &) [inline, protected]`

6.198.3.19 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive () throw (cms::CMSException) [virtual]`

Performs a synchronous read from the default destination.

Returns

the message

Exceptions

cms::CMSEException (p. 1074) thrown if an error occurs

6.198.3.20 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (const std::string & destinationName) throw (cms::CMSEException)`
[virtual]

Performs a synchronous read from the specified destination.

Parameters

destinationName the name of the destination to receive on (will be resolved to destination internally).

Returns

the message

Exceptions

cms::CMSEException (p. 1074) thrown if an error occurs

6.198.3.21 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (cms::Destination * destination) throw (cms::CMSEException)`
[virtual]

Performs a synchronous read from the specified destination.

Parameters

destination the destination to receive on

Returns

the message

Exceptions

cms::CMSEException (p. 1074) thrown if an error occurs

6.198.3.22 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (const std::string & destinationName, const std::string & selector) throw (cms::CMSEException)` [virtual]

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Parameters

destinationName the name of the destination to receive on (will be resolved to destination internally).

selector the selector expression.

Returns

the message

Exceptions

cms::CMSEException (p. 1074) thrown if an error occurs

6.198.3.23 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected
(cms::Destination * destination, const std::string & selector) throw
(cms::CMSEException) [virtual]`

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Parameters

destination the destination to receive on.

selector the selector expression.

Returns

the message

Exceptions

cms::CMSEException (p. 1074) thrown if an error occurs

6.198.3.24 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected
(const std::string & selector) throw (cms::CMSEException)
[virtual]`

Performs a synchronous read consuming only messages identified by the given selector.

Parameters

selector the selector expression.

Returns

the message

Exceptions

cms::CMSEException (p. 1074) thrown if an error occurs

6.198.3.25 `virtual void activemq::cmsutil::CmsTemplate::send (const std::string & destinationName, MessageCreator * messageCreator) throw (cms::CMSEException) [virtual]`

Convenience method for sending a message to the specified destination.

Parameters

destinationName The name of the destination to send to.

messageCreator Responsible for creating the message to be sent

Exceptions

cms::CMSEException (p. 1074) thrown if an error occurs.

6.198.3.26 `virtual void activemq::cmsutil::CmsTemplate::send (MessageCreator * messageCreator) throw (cms::CMSEException) [virtual]`

Convenience method for sending a message to the default destination.

Parameters

messageCreator Responsible for creating the message to be sent

Exceptions

cms::CMSEException (p. 1074) thrown if an error occurs.

6.198.3.27 `virtual void activemq::cmsutil::CmsTemplate::send (cms::Destination * dest, MessageCreator * messageCreator) throw (cms::CMSEException) [virtual]`

Convenience method for sending a message to the specified destination.

Parameters

dest The destination to send to

messageCreator Responsible for creating the message to be sent

Exceptions

cms::CMSEException (p. 1074) thrown if an error occurs.

6.198.3.28 `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestination (cms::Destination * defaultDestination) [inline, virtual]`

Sets the destination object to be used by default for send/receive operations.

If no default destination is provided, the `defaultDestinationName` property is used to resolve this default destination for send/receive operations.

Parameters

defaultDestination the default destination

6.198.3.29 `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestinationName (const std::string & defaultDestinationName) [inline, virtual]`

Sets the name of the default destination to be used from send/receive operations.

Calling this method will set the `defaultDestination` property to NULL. The destination type (topic/queue) is determined by the `pubSubDomain` property.

Parameters

defaultDestinationName the name of the destination for send/receive to by default.

6.198.3.30 `virtual void activemq::cmsutil::CmsTemplate::setDeliveryMode (int deliveryMode) [inline, virtual]`

Set the delivery mode to use when sending a message.

Default is the Message default: "PERSISTENT".

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

Parameters

deliveryMode the delivery mode to use

See also

`isExplicitQosEnabled` (p. 1090)

6.198.3.31 `virtual void activemq::cmsutil::CmsTemplate::setDeliveryPersistent (bool deliveryPersistent) [inline, virtual]`

Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").

This will set the delivery mode accordingly, to either "PERSISTENT" or "NON_PERSISTENT".

Default it "true" aka delivery mode "PERSISTENT".

See also

`setDeliveryMode(int)` (p. 1094)

6.198.3.32 `virtual void activemq::cmsutil::CmsTemplate::setExplicitQosEnabled (bool explicitQosEnabled) [inline, virtual]`

Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.

See also

`setDeliveryMode` (p. 1094)

`setPriority` (p. 1095)

`setTimeToLive` (p. 1095)

- 6.198.3.33** `virtual void activemq::cmsutil::CmsTemplate::setMessageIdEnabled (bool messageIdEnabled) [inline, virtual]`
- 6.198.3.34** `virtual void activemq::cmsutil::CmsTemplate::setMessageTimestampEnabled (bool messageTimestampEnabled) [inline, virtual]`
- 6.198.3.35** `virtual void activemq::cmsutil::CmsTemplate::setNoLocal (bool noLocal) [inline, virtual]`
- 6.198.3.36** `virtual void activemq::cmsutil::CmsTemplate::setPriority (int priority) [inline, virtual]`

Set the priority of a message when sending.

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

See also

`isExplicitQosEnabled` (p. 1090)

- 6.198.3.37** `virtual void activemq::cmsutil::CmsTemplate::setPubSubDomain (bool pubSubDomain) [inline, virtual]`

Indicates whether the default destination is a topic (true) or a queue (false).

Calling this method will set the `defaultDestination` property to NULL.

Parameters

pubSubDomain indicates whether to use pub-sub messaging (topics).

Reimplemented from `activemq::cmsutil::CmsDestinationAccessor` (p. 1074).

References `activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain()`.

- 6.198.3.38** `virtual void activemq::cmsutil::CmsTemplate::setReceiveTimeout (long long receiveTimeout) [inline, virtual]`
- 6.198.3.39** `virtual void activemq::cmsutil::CmsTemplate::setTimeToLive (long long timeToLive) [inline, virtual]`

Set the time-to-live of the message when sending.

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

Parameters

timeToLive the message's lifetime (in milliseconds)

See also

`isExplicitQosEnabled` (p. 1090)

6.198.4 Friends And Related Function Documentation

6.198.4.1 friend class `ProducerExecutor` [friend]

6.198.4.2 friend class `ReceiveExecutor` [friend]

6.198.4.3 friend class `ResolveProducerExecutor` [friend]

6.198.4.4 friend class `ResolveReceiveExecutor` [friend]

6.198.4.5 friend class `SendExecutor` [friend]

6.198.5 Field Documentation

6.198.5.1 `const int activemq::cmsutil::CmsTemplate::DEFAULT_PRIORITY`
[static]

Default message priority.

6.198.5.2 `const long long activemq::cmsutil::CmsTemplate::DEFAULT_TIME_TO_LIVE` [static]

My default, messages should live forever.

6.198.5.3 `const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_INDEFINITE_WAIT` [static]

Timeout value indicating a blocking receive without timeout.

6.198.5.4 `const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_NO_WAIT` [static]

Timeout value indicating that a receive operation should check if a message is immediately available without blocking.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.199 code Struct Reference

```
#include <src/main/decaf/internal/util/zip/inftrees.h>
```

Data Fields

- unsigned char `op`
- unsigned char `bits`
- unsigned short `val`

6.199.1 Field Documentation

6.199.1.1 unsigned char code::bits

6.199.1.2 unsigned char code::op

6.199.1.3 unsigned short code::val

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/inftrees.h`

6.200 decaf::util::Collection< E > Class Template Reference

The root interface in the collection hierarchy.

```
#include <src/main/decaf/util/Collection.h>
```

Inheritance diagram for `decaf::util::Collection< E >`:

Public Member Functions

- virtual `~Collection()`
- virtual `bool add(const E &value)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)`
Returns true if this collection changed as a result of the call.
- virtual `bool addAll(const Collection< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)`
Adds all of the elements in the specified collection to this collection.
- virtual `void clear()=0 throw (lang::exceptions::UnsupportedOperationException)`
Removes all of the elements from this collection (optional operation).
- virtual `bool contains(const E &value) const =0 throw (lang::Exception)`
Returns true if this collection contains the specified element.
- virtual `bool containsAll(const Collection< E > &collection) const =0 throw (lang::Exception)`
Returns true if this collection contains all of the elements in the specified collection.
- virtual `bool equals(const Collection< E > &value) const =0`
Compares the passed collection to this one, if they contain the same elements, i.e.
- virtual `bool isEmpty() const =0`
- virtual `bool remove(const E &value)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)`
Removes a single instance of the specified element from the collection.

- virtual bool **removeAll** (const **Collection**< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

Removes all this collection's elements that are also contained in the specified collection (optional operation).

- virtual bool **retainAll** (const **Collection**< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

Retains only the elements in this collection that are contained in the specified collection (optional operation).

- virtual std::size_t **size** () const =0

Returns the number of elements in this collection.

- virtual std::vector< E > **toArray** () const =0

Returns an array containing all of the elements in this collection.

6.200.1 Detailed Description

template<typename E> class decaf::util::Collection< E >

The root interface in the collection hierarchy. A collection represents a group of objects, known as its elements. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.

All general-purpose **Collection** (p. 1097) implementation classes (which typically implement **Collection** (p. 1097) indirectly through one of its subinterfaces) should provide two "standard" constructors: a void (no arguments) constructor, which creates an empty collection, and a constructor with a single argument of type **Collection** (p. 1097), which creates a new collection with the same elements as its argument. In effect, the latter constructor allows the user to copy any collection, producing an equivalent collection of the desired implementation type. There is no way to enforce this convention (as interfaces cannot contain constructors) but all of the general-purpose **Collection** (p. 1097) implementations in the Decaf platform libraries comply.

The "destructive" methods contained in this interface, that is, the methods that modify the collection on which they operate, are specified to throw **UnsupportedOperationException** if this collection does not support the operation. If this is the case, these methods may, but are not required to, throw an **UnsupportedOperationException** if the invocation would have no effect on the collection. For example, invoking the **addAll(Collection)** method on an unmodifiable collection may, but is not required to, throw the exception if the collection to be added is empty.

Many methods in Collections Framework interfaces are defined in terms of the **equals** method. For example, the specification for the **contains(Object o)** method says: "returns true if and only if this collection contains at least one element e such that (o==null ? e==null : o.equals(e))."

Since

1.0

6.200.2 Constructor & Destructor Documentation

6.200.2.1 `template<typename E> virtual decaf::util::Collection< E >::~~Collection () [inline, virtual]`

6.200.3 Member Function Documentation

6.200.3.1 `template<typename E> virtual bool decaf::util::Collection< E >::add (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [pure virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p.1097) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value - reference to the element to add.

Returns

true if the element was added

Exceptions

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implemented in **decaf::util::AbstractQueue< E >** (p.159), **decaf::util::PriorityQueue< E >** (p.2835), **decaf::util::StlList< E >** (p.3363), **decaf::util::StlSet< E >** (p.3392), **decaf::util::StlList< cms::MessageConsumer * >** (p.3363), **decaf::util::StlList< CompositeTask * >** (p.3363), **decaf::util::StlList< URI >** (p.3363), **decaf::util::StlList< Pointer< DestinationInfo > >** (p.3363), **decaf::util::StlList< PrimitiveValueNode >** (p.3363), **decaf::util::StlList< Pointer< Command > >** (p.3363), **decaf::util::StlList< Pointer< BackupTransport > >** (p.3363), **decaf::util::StlList< cms::MessageProducer * >** (p.3363), **decaf::util::StlList< cms::Destination * >** (p.3363), **decaf::util::StlList< cms::Session * >** (p.3363), **decaf::util::StlList< cms::Connection * >** (p.3363),

`decaf::util::StlSet< transport::TransportListener * >` (p. 3392), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3392), `decaf::util::StlSet< Resource * >` (p. 3392), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3392).

```
6.200.3.2  template<typename E> virtual bool decaf::util::Collection<
           E >::addAll ( const Collection< E > & collection ) throw
           ( lang::exceptions::UnsupportedOperationException,
             lang::exceptions::IllegalArgumentException,
             lang::exceptions::IllegalStateException ) [pure
           virtual]
```

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

collection - **Collection** (p. 1097) whose elements are added to this one.

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implemented in `decaf::util::AbstractCollection< E >` (p. 147), `decaf::util::AbstractQueue< E >` (p. 160), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 147), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 147), `decaf::util::AbstractCollection< Resource * >` (p. 147), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 147), `decaf::util::AbstractCollection< CompositeTask * >` (p. 147), `decaf::util::AbstractCollection< URI >` (p. 147), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 147), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 147), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 147), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 147), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 147), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 147), `decaf::util::AbstractCollection< cms::Destination * >` (p. 147), `decaf::util::AbstractCollection< cms::Session * >` (p. 147), and `decaf::util::AbstractCollection< cms::Connection * >` (p. 147).

```
6.200.3.3  template<typename E> virtual void decaf::util::Collection< E >::clear (
           ) throw ( lang::exceptions::UnsupportedOperationException ) [pure
           virtual]
```

Removes all of the elements from this collection (optional operation).

This collection will be empty after this method returns unless it throws an exception.

Exceptions

UnsupportedOperationException

Implemented in `decaf::util::AbstractCollection< E >` (p. 147), `decaf::util::AbstractQueue< E >` (p. 160), `decaf::util::concurrent::SynchronousQueue< E >` (p. 3480), `decaf::util::PriorityQueue< E >` (p. 2836), `decaf::util::StlList< E >` (p. 3364), `decaf::util::StlSet< E >` (p. 3393), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 147), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 147), `decaf::util::AbstractCollection< Resource * >` (p. 147), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 147), `decaf::util::AbstractCollection< CompositeTask * >` (p. 147), `decaf::util::AbstractCollection< URI >` (p. 147), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 147), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 147), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 147), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 147), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 147), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 147), `decaf::util::AbstractCollection< cms::Destination * >` (p. 147), `decaf::util::AbstractCollection< cms::Session * >` (p. 147), `decaf::util::AbstractCollection< cms::Connection * >` (p. 147), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3364), `decaf::util::StlList< CompositeTask * >` (p. 3364), `decaf::util::StlList< URI >` (p. 3364), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3364), `decaf::util::StlList< PrimitiveValueNode >` (p. 3364), `decaf::util::StlList< Pointer< Command > >` (p. 3364), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3364), `decaf::util::StlList< cms::MessageProducer * >` (p. 3364), `decaf::util::StlList< cms::Destination * >` (p. 3364), `decaf::util::StlList< cms::Session * >` (p. 3364), `decaf::util::StlList< cms::Connection * >` (p. 3364), `decaf::util::StlSet< transport::TransportListener * >` (p. 3393), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3393), `decaf::util::StlSet< Resource * >` (p. 3393), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3393).

6.200.3.4 `template<typename E> virtual bool decaf::util::Collection< E >::contains (const E & value) const throw (lang::Exception) [pure virtual]`

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element *e* such that (*o*==null ? *e*==null : *o.equals(e)*).

Parameters

value - value to check for presence in the collection

Returns

true if there is at least one of the elements in the collection

Exceptions

Exception

Implemented in `decaf::util::AbstractCollection< E >` (p. 148), `decaf::util::StlList< E >` (p. 3365), `decaf::util::StlSet< E >` (p. 3394), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 148), `decaf::util::AbstractCollection<`

Pointer< Synchronization > > (p. 148), decaf::util::AbstractCollection< Resource * > (p. 148), decaf::util::AbstractCollection< cms::MessageConsumer * > (p. 148), decaf::util::AbstractCollection< CompositeTask * > (p. 148), decaf::util::AbstractCollection< URI > (p. 148), decaf::util::AbstractCollection< ActiveMQSession * > (p. 148), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 148), decaf::util::AbstractCollection< PrimitiveValueNode > > (p. 148), decaf::util::AbstractCollection< Pointer< Command > > (p. 148), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 148), decaf::util::AbstractCollection< cms::MessageProducer * > (p. 148), decaf::util::AbstractCollection< cms::Destination * > (p. 148), decaf::util::AbstractCollection< cms::Session * > (p. 148), decaf::util::AbstractCollection< cms::Connection * > (p. 148), decaf::util::StlList< cms::MessageConsumer * > (p. 3365), decaf::util::StlList< CompositeTask * > (p. 3365), decaf::util::StlList< URI > (p. 3365), decaf::util::StlList< Pointer< DestinationInfo > > (p. 3365), decaf::util::StlList< PrimitiveValueNode > > (p. 3365), decaf::util::StlList< Pointer< Command > > (p. 3365), decaf::util::StlList< Pointer< BackupTransport > > (p. 3365), decaf::util::StlList< cms::MessageProducer * > (p. 3365), decaf::util::StlList< cms::Destination * > (p. 3365), decaf::util::StlList< cms::Session * > (p. 3365), decaf::util::StlList< cms::Connection * > (p. 3365), decaf::util::StlSet< transport::TransportListener * > (p. 3394), decaf::util::StlSet< Pointer< Synchronization > > (p. 3394), decaf::util::StlSet< Resource * > (p. 3394), and decaf::util::StlSet< ActiveMQSession * > (p. 3394).

6.200.3.5 template<typename E> virtual bool decaf::util::Collection< E >::containsAll (const Collection< E > & *collection*) const throw (lang::Exception) [pure virtual]

Returns true if this collection contains all of the elements in the specified collection.

Parameters

collection - Collection (p. 1097) to compare to this one.

Exceptions

Exception

Implemented in decaf::util::AbstractCollection< E > (p. 149), decaf::util::concurrent::SynchronousQueue< E > (p. 3480), decaf::util::AbstractCollection< transport::TransportListener * > (p. 149), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 149), decaf::util::AbstractCollection< Resource * > (p. 149), decaf::util::AbstractCollection< cms::MessageConsumer * > (p. 149), decaf::util::AbstractCollection< CompositeTask * > (p. 149), decaf::util::AbstractCollection< URI > (p. 149), decaf::util::AbstractCollection< ActiveMQSession * > (p. 149), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 149), decaf::util::AbstractCollection< PrimitiveValueNode > > (p. 149), decaf::util::AbstractCollection< Pointer< Command > > (p. 149), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 149), decaf::util::AbstractCollection< cms::MessageProducer * > (p. 149), decaf::util::AbstractCollection< cms::Destination * > (p. 149), decaf::util::AbstractCollection< cms::Session * > (p. 149), and decaf::util::AbstractCollection< cms::Connection * > (p. 149).

6.200.3.6 `template<typename E> virtual bool decaf::util::Collection< E >::equals (const Collection< E > & value) const [pure virtual]`

Compares the passed collection to this one, if they contain the same elements, i.e.

all their elements are equivalent, then it returns true.

Returns

true if the Collections contain the same elements.

Implemented in `decaf::util::AbstractCollection< E >` (p. 149), `decaf::util::concurrent::SynchronousQueue< E >` (p. 3482), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 149), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 149), `decaf::util::AbstractCollection< Resource * >` (p. 149), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 149), `decaf::util::AbstractCollection< CompositeTask * >` (p. 149), `decaf::util::AbstractCollection< URI >` (p. 149), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 149), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 149), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 149), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 149), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 149), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 149), `decaf::util::AbstractCollection< cms::Destination * >` (p. 149), `decaf::util::AbstractCollection< cms::Session * >` (p. 149), and `decaf::util::AbstractCollection< cms::Connection * >` (p. 149).

6.200.3.7 `template<typename E> virtual bool decaf::util::Collection< E >::isEmpty () const [pure virtual]`

Returns

true if this collection contains no elements.

Implemented in `decaf::util::AbstractCollection< E >` (p. 150), `decaf::util::concurrent::SynchronousQueue< E >` (p. 3482), `decaf::util::StlList< E >` (p. 3366), `decaf::util::StlSet< E >` (p. 3394), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 150), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 150), `decaf::util::AbstractCollection< Resource * >` (p. 150), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 150), `decaf::util::AbstractCollection< CompositeTask * >` (p. 150), `decaf::util::AbstractCollection< URI >` (p. 150), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 150), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 150), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 150), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 150), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 150), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 150), `decaf::util::AbstractCollection< cms::Destination * >` (p. 150), `decaf::util::AbstractCollection< cms::Session * >` (p. 150), `decaf::util::AbstractCollection< cms::Connection * >` (p. 150), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3366), `decaf::util::StlList< CompositeTask * >` (p. 3366), `decaf::util::StlList< URI >` (p. 3366), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3366), `decaf::util::StlList< PrimitiveValueNode >` (p. 3366), `decaf::util::StlList< Pointer< Command > >` (p. 3366), `decaf::util::StlList< Pointer<`

BackupTransport > > (p. 3366), decaf::util::StlList< cms::MessageProducer * > (p. 3366), decaf::util::StlList< cms::Destination * > (p. 3366), decaf::util::StlList< cms::Session * > (p. 3366), decaf::util::StlList< cms::Connection * > (p. 3366), decaf::util::StlSet< transport::TransportListener * > (p. 3394), decaf::util::StlSet< Pointer< Synchronization > > (p. 3394), decaf::util::StlSet< Resource * > (p. 3394), and decaf::util::StlSet< ActiveMQSession * > (p. 3394).

6.200.3.8 `template<typename E> virtual bool decaf::util::Collection< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [pure virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (*o*==null ? *e*==null : *o*.equals(*e*)), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

value - reference to the element to remove.

Returns

true if the collection was changed

Exceptions

UnsupportedOperationException

IllegalArgumentException

Implemented in decaf::util::AbstractCollection< E > (p. 151), decaf::util::PriorityQueue< E > (p. 2838), decaf::util::StlList< E > (p. 3368), decaf::util::StlSet< E > (p. 3394), decaf::util::AbstractCollection< transport::TransportListener * > (p. 151), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 151), decaf::util::AbstractCollection< Resource * > (p. 151), decaf::util::AbstractCollection< cms::MessageConsumer * > (p. 151), decaf::util::AbstractCollection< CompositeTask * > (p. 151), decaf::util::AbstractCollection< URI > (p. 151), decaf::util::AbstractCollection< ActiveMQSession * > (p. 151), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 151), decaf::util::AbstractCollection< PrimitiveValueNode > (p. 151), decaf::util::AbstractCollection< Pointer< Command > > (p. 151), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 151), decaf::util::AbstractCollection< cms::MessageProducer * > (p. 151), decaf::util::AbstractCollection< cms::Destination * > (p. 151), decaf::util::AbstractCollection< cms::Session * > (p. 151), decaf::util::AbstractCollection< cms::Connection * > (p. 151), decaf::util::StlList< cms::MessageConsumer * > (p. 3368), decaf::util::StlList< CompositeTask * > (p. 3368), decaf::util::StlList< URI > (p. 3368), decaf::util::StlList< Pointer< DestinationInfo > > (p. 3368), decaf::util::StlList< PrimitiveValueNode > (p. 3368), decaf::util::StlList< Pointer< Command > > (p. 3368), decaf::util::StlList< Pointer< BackupTransport > > (p. 3368), decaf::util::StlList< cms::MessageProducer * > (p. 3368), decaf::util::StlList< cms::Destination * > (p. 3368), decaf::util::StlList< cms::Session * > (p. 3368), decaf::util::StlList< cms::Connection * > (p. 3368), decaf::util::StlSet< transport::TransportListener * > (p. 3394), decaf::util::StlSet<

`Pointer< Synchronization > >` (p. 3394), `decaf::util::StlSet< Resource * >` (p. 3394), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3394).

6.200.3.9 `template<typename E> virtual bool decaf::util::Collection< E >::removeAll (const Collection< E > & collection)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException) [pure virtual]`

Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

Parameters

collection - The **Collection** (p. 1097) whose elements are to be removed

Returns

true if the collection changed as a result of this call

Exceptions

UnsupportedOperationException

IllegalArgumentException

Implemented in `decaf::util::AbstractCollection< E >` (p. 152), `decaf::util::AbstractSet< E >` (p. 163), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 152), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 152), `decaf::util::AbstractCollection< Resource * >` (p. 152), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 152), `decaf::util::AbstractCollection< CompositeTask * >` (p. 152), `decaf::util::AbstractCollection< URI >` (p. 152), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 152), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 152), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 152), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 152), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 152), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 152), `decaf::util::AbstractCollection< cms::Destination * >` (p. 152), `decaf::util::AbstractCollection< cms::Session * >` (p. 152), `decaf::util::AbstractCollection< cms::Connection * >` (p. 152), `decaf::util::AbstractSet< transport::TransportListener * >` (p. 163), `decaf::util::AbstractSet< Pointer< Synchronization > >` (p. 163), `decaf::util::AbstractSet< Resource * >` (p. 163), and `decaf::util::AbstractSet< ActiveMQSession * >` (p. 163).

6.200.3.10 `template<typename E> virtual bool decaf::util::Collection< E >::retainAll (const Collection< E > & collection)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException) [pure virtual]`

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

Parameters

collection - The **Collection** (p. 1097) whose elements are to be retained

Returns

true if the collection changed as a result of this call

Exceptions

UnsupportedOperationException

IllegalArgumentException

Implemented in **decaf::util::AbstractCollection< E >** (p. 153), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 153), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 153), **decaf::util::AbstractCollection< Resource * >** (p. 153), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 153), **decaf::util::AbstractCollection< CompositeTask * >** (p. 153), **decaf::util::AbstractCollection< URI >** (p. 153), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 153), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 153), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 153), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 153), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 153), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 153), **decaf::util::AbstractCollection< cms::Destination * >** (p. 153), **decaf::util::AbstractCollection< cms::Session * >** (p. 153), and **decaf::util::AbstractCollection< cms::Connection * >** (p. 153).

6.200.3.11 `template<typename E> virtual std::size_t decaf::util::Collection< E >::size () const [pure virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX_VALUE elements, returns Integer.MAX_VALUE.

Returns

the number of elements in this collection

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3486), **decaf::util::PriorityQueue< E >** (p. 2839), **decaf::util::StlList< E >** (p. 3369), **decaf::util::StlSet< E >** (p. 3395), **decaf::util::StlList< cms::MessageConsumer * >** (p. 3369), **decaf::util::StlList< CompositeTask * >** (p. 3369), **decaf::util::StlList< URI >** (p. 3369), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3369), **decaf::util::StlList< PrimitiveValueNode >** (p. 3369), **decaf::util::StlList< Pointer< Command > >** (p. 3369), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3369), **decaf::util::StlList< cms::MessageProducer * >** (p. 3369), **decaf::util::StlList< cms::Destination * >** (p. 3369), **decaf::util::StlList< cms::Session * >** (p. 3369), **decaf::util::StlList< cms::Connection * >** (p. 3369), **decaf::util::StlSet< transport::TransportListener * >** (p. 3395), **decaf::util::StlSet< Pointer< Synchronization**

> > (p. 3395), `decaf::util::StlSet< Resource * >` (p. 3395), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3395).

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::equals()`, `decaf::util::AbstractCollection< cms::Connection * >::isEmpty()`, `decaf::util::AbstractSet< ActiveMQSession * >::removeAll()`, and `decaf::util::AbstractCollection< cms::Connection * >::toArray()`.

6.200.3.12 `template<typename E> virtual std::vector<E> decaf::util::Collection< E >::toArray () const [pure virtual]`

Returns an array containing all of the elements in this collection.

If the collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

This method acts as bridge between array-based and collection-based APIs.

Returns

an array of the elements in this collection.

Implemented in `decaf::util::AbstractCollection< E >` (p. 153), `decaf::util::concurrent::SynchronousQueue< E >` (p. 3486), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 153), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 153), `decaf::util::AbstractCollection< Resource * >` (p. 153), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 153), `decaf::util::AbstractCollection< CompositeTask * >` (p. 153), `decaf::util::AbstractCollection< URI >` (p. 153), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 153), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 153), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 153), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 153), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 153), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 153), `decaf::util::AbstractCollection< cms::Destination * >` (p. 153), `decaf::util::AbstractCollection< cms::Session * >` (p. 153), and `decaf::util::AbstractCollection< cms::Connection * >` (p. 153).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Collection.h`

6.201 activemq::commands::Command Class Reference

```
#include <src/main/activemq/commands/Command.h>
```

Inheritance diagram for `activemq::commands::Command`:

Public Member Functions

- `virtual ~Command ()`

- virtual void **setCommandId** (int id)=0
*Sets the **Command** (p. 1107) Id of this **Message** (p. 2358).*
- virtual int **getCommandId** () const =0
*Gets the **Command** (p. 1107) Id of this **Message** (p. 2358).*
- virtual void **setResponseRequired** (const bool required)=0
*Set if this **Message** (p. 2358) requires a **Response** (p. 3076).*
- virtual bool **isResponseRequired** () const =0
*Is a **Response** (p. 3076) required for this **Command** (p. 1107).*
- virtual std::string **toString** () const =0
Returns a provider-specific string that provides information about the contents of the command.
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor)=0 throw (**exceptions::ActiveMQException**)
*Allows a **Visitor** to visit this command and return a response to the command based on the command type being visited.*
- virtual bool **isConnectionInfo** () const =0
- virtual bool **isConsumerInfo** () const =0
- virtual bool **isBrokerInfo** () const =0
- virtual bool **isKeepAliveInfo** () const =0
- virtual bool **isMessage** () const =0
- virtual bool **isMessageAck** () const =0
- virtual bool **isMessageDispatch** () const =0
- virtual bool **isMessageDispatchNotification** () const =0
- virtual bool **isProducerAck** () const =0
- virtual bool **isProducerInfo** () const =0
- virtual bool **isResponse** () const =0
- virtual bool **isRemoveInfo** () const =0
- virtual bool **isRemoveSubscriptionInfo** () const =0
- virtual bool **isShutdownInfo** () const =0
- virtual bool **isTransactionInfo** () const =0
- virtual bool **isWireFormatInfo** () const =0

6.201.1 Constructor & Destructor Documentation

- 6.201.1.1 virtual **activemq::commands::Command::~Command** () [inline, virtual]

6.201.2 Member Function Documentation

- 6.201.2.1 virtual int **activemq::commands::Command::getCommandId** () const [pure virtual]

Gets the **Command** (p. 1107) Id of this **Message** (p. 2358).

Returns

Command (p. 1107) Id

Implemented in **activemq::commands::BaseCommand** (p. 697).

6.201.2.2 **virtual bool activemq::commands::Command::isBrokerInfo () const**
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 698), and **activemq::commands::BrokerInfo** (p. 828).

6.201.2.3 **virtual bool activemq::commands::Command::isConnectionInfo () const**
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 698), and **activemq::commands::ConnectionInfo** (p. 1261).

6.201.2.4 **virtual bool activemq::commands::Command::isConsumerInfo () const**
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 698), and **activemq::commands::ConsumerInfo** (p. 1362).

6.201.2.5 **virtual bool activemq::commands::Command::isKeepAliveInfo () const**
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 698), and **activemq::commands::KeepAliveInfo** (p. 2124).

6.201.2.6 **virtual bool activemq::commands::Command::isMessage () const**
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 698), and **activemq::commands::Message** (p. 2369).

6.201.2.7 **virtual bool activemq::commands::Command::isMessageAck () const**
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 698), and **activemq::commands::MessageAck** (p. 2397).

6.201.2.8 **virtual bool activemq::commands::Command::isMessageDispatch () const**
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 698), and **activemq::commands::MessageDispatch** (p. 2429).

6.201.2.9 `virtual bool activemq::commands::Command::isMessageDispatchNotification () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 699), and `activemq::commands::MessageDispatchNotification` (p. 2465).

6.201.2.10 `virtual bool activemq::commands::Command::isProducerAck () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 699), and `activemq::commands::ProducerAck` (p. 2841).

6.201.2.11 `virtual bool activemq::commands::Command::isProducerInfo () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 699), and `activemq::commands::ProducerInfo` (p. 2900).

6.201.2.12 `virtual bool activemq::commands::Command::isRemoveInfo () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 699), and `activemq::commands::RemoveInfo` (p. 2990).

6.201.2.13 `virtual bool activemq::commands::Command::isRemoveSubscriptionInfo () const` [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 699), and `activemq::commands::RemoveSubscriptionInfo` (p. 3018).

6.201.2.14 `virtual bool activemq::commands::Command::isResponse () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 699), and `activemq::commands::Response` (p. 3078).

6.201.2.15 `virtual bool activemq::commands::Command::isResponseRequired () const` [pure virtual]

Is a **Response** (p. 3076) required for this **Command** (p. 1107).

Returns

true if a response is required.

Implemented in `activemq::commands::BaseCommand` (p. 699).

6.201.2.16 `virtual bool activemq::commands::Command::isShutdownInfo ()`
`const [pure virtual]`

Implemented in `activemq::commands::BaseCommand` (p. 700), and `activemq::commands::ShutdownInfo` (p. 3251).

6.201.2.17 `virtual bool activemq::commands::Command::isTransactionInfo ()`
`const [pure virtual]`

Implemented in `activemq::commands::BaseCommand` (p. 700), and `activemq::commands::TransactionInfo` (p. 3597).

6.201.2.18 `virtual bool activemq::commands::Command::isWireFormatInfo ()`
`const [pure virtual]`

Implemented in `activemq::commands::BaseCommand` (p. 700), and `activemq::commands::WireFormatInfo` (p. 3724).

6.201.2.19 `virtual void activemq::commands::Command::setCommandId (int id`
`) [pure virtual]`

Sets the **Command** (p. 1107) Id of this **Message** (p. 2358).

Parameters

id **Command** (p. 1107) Id

Implemented in `activemq::commands::BaseCommand` (p. 700).

6.201.2.20 `virtual void activemq::commands::Command::setResponseRequired (`
`const bool required) [pure virtual]`

Set if this **Message** (p. 2358) requires a **Response** (p. 3076).

Parameters

required true if response is required

Implemented in `activemq::commands::BaseCommand` (p. 700).

6.201.2.21 `virtual std::string activemq::commands::Command::toString () const`
`[pure virtual]`

Returns a provider-specific string that provides information about the contents of the command.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 767).

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 170), `activemq::commands::ActiveMQBytesMessage` (p. 205), `activemq::commands::ActiveMQMapMessage` (p. 328), `activemq::commands::ActiveMQMessage` (p. 355), `activemq::commands::ActiveMQObjectMessage` (p. 398), `activemq::commands::ActiveMQStreamMessage` (p. 495),

activemq::commands::ActiveMQTextMessage (p. 609), **activemq::commands::BaseCommand** (p. 700), **activemq::commands::BrokerInfo** (p. 829), **activemq::commands::ConnectionControl** (p. 1175), **activemq::commands::ConnectionError** (p. 1203), **activemq::commands::ConnectionInfo** (p. 1262), **activemq::commands::ConsumerControl** (p. 1305), **activemq::commands::ConsumerInfo** (p. 1363), **activemq::commands::ControlCommand** (p. 1392), **activemq::commands::DataArrayResponse** (p. 1425), **activemq::commands::DataResponse** (p. 1479), **activemq::commands::DestinationInfo** (p. 1617), **activemq::commands::ExceptionResponse** (p. 1722), **activemq::commands::FlushCommand** (p. 1814), **activemq::commands::IntegerResponse** (p. 1958), **activemq::commands::KeepAliveInfo** (p. 2124), **activemq::commands::Message** (p. 2372), **activemq::commands::MessageAck** (p. 2398), **activemq::commands::MessageDispatch** (p. 2430), **activemq::commands::MessageDispatchNotification** (p. 2465), **activemq::commands::MessagePull** (p. 2567), **activemq::commands::ProducerAck** (p. 2842), **activemq::commands::ProducerInfo** (p. 2901), **activemq::commands::RemoveInfo** (p. 2990), **activemq::commands::RemoveSubscriptionInfo** (p. 3018), **activemq::commands::ReplayCommand** (p. 3045), **activemq::commands::Response** (p. 3079), **activemq::commands::SessionInfo** (p. 3191), **activemq::commands::ShutdownInfo** (p. 3251), **activemq::commands::TransactionInfo** (p. 3597), and **activemq::commands::WireFormatInfo** (p. 3727).

6.201.2.22 `virtual decaf::lang::Pointer<commands::Command> activemq::commands::Command::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [pure virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implemented in **activemq::commands::BrokerError** (p. 796), **activemq::commands::BrokerInfo** (p. 830), **activemq::commands::ConnectionControl** (p. 1176), **activemq::commands::ConnectionError** (p. 1203), **activemq::commands::ConnectionInfo** (p. 1262), **activemq::commands::ConsumerControl** (p. 1306), **activemq::commands::ConsumerInfo** (p. 1364), **activemq::commands::ControlCommand** (p. 1393), **activemq::commands::DestinationInfo** (p. 1617), **activemq::commands::FlushCommand** (p. 1814), **activemq::commands::KeepAliveInfo** (p. 2125), **activemq::commands::Message** (p. 2373), **activemq::commands::MessageAck** (p. 2398), **activemq::commands::MessageDispatch** (p. 2430), **activemq::commands::MessageDispatchNotification** (p. 2466), **activemq::commands::MessagePull** (p. 2567), **activemq::commands::ProducerAck** (p. 2842), **activemq::commands::ProducerInfo** (p. 2901), **activemq::commands::RemoveInfo** (p. 2991), **activemq::commands::RemoveSubscriptionInfo** (p. 3018), **activemq::commands::ReplayCommand** (p. 3045), **activemq::commands::Response** (p. 3079), **activemq::commands::SessionInfo** (p. 3191), **activemq::commands::ShutdownInfo** (p. 3252), **activemq::commands::TransactionInfo** (p. 3598), and **activemq::commands::WireFormatInfo** (p. 3727).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/Command.h`

6.202 `activemq::state::CommandVisitor` Class Reference

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

```
#include <src/main/activemq/state/CommandVisitor.h>
```

Inheritance diagram for `activemq::state::CommandVisitor`:

Public Member Functions

- `virtual ~CommandVisitor ()`
- `virtual decaf::lang::Pointer< commands::Command > processTransactionInfo (commands::TransactionInfo *info)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveInfo (commands::RemoveInfo *info)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processConnectionInfo (commands::ConnectionInfo *info)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processSessionInfo (commands::SessionInfo *info)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processProducerInfo (commands::ProducerInfo *info)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processConsumerInfo (commands::ConsumerInfo *info)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveConnection (commands::ConnectionId *id)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveSession (commands::SessionId *id)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveProducer (commands::ProducerId *id)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveConsumer (commands::ConsumerId *id)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processDestinationInfo (commands::DestinationInfo *info)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveDestination (commands::DestinationInfo *info)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo (commands::RemoveSubscriptionInfo *info)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processMessage (commands::Message *send)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processMessageAck (commands::MessageAck *ack)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processMessagePull (commands::MessagePull *pull)=0 throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processBeginTransaction (commands::TransactionInfo *info)=0 throw (exceptions::ActiveMQException)`

- virtual **decaf::lang::Pointer**< **commands::Command** > **processPrepareTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processCommitTransactionOnePhase** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processcommitTransactionTwoPhase** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRollbackTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processFlushCommand** (**commands::FlushCommand** *command)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** *notification)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerAck** (**commands::ProducerAck** *ack)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** *dispatch)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processControlCommand** (**commands::ControlCommand** *command)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionError** (**commands::ConnectionError** *error)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** *control)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** *control)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerError** (**commands::BrokerError** *error)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processReplayCommand** (**commands::ReplayCommand** *replay)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processResponse** (**commands::Response** *response)=0 throw (exceptions::ActiveMQException)

6.202.1 Detailed Description

Interface for an Object that can visit the various Command Objects that are sent from and to this client. The Commands themselves implement a `visit` method that is called with an instance of this interface and each one then call the appropriate `processXXX` method.

Since

3.0

6.202.2 Constructor & Destructor Documentation

6.202.2.1 `virtual activemq::state::CommandVisitor::~~CommandVisitor ()`
[inline, virtual]

6.202.3 Member Function Documentation

6.202.3.1 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitor::processBeginTransaction`
(`commands::TransactionInfo * info`) throw (`exceptions::ActiveMQException`) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1296).

6.202.3.2 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitor::processBrokerError`
(`commands::BrokerError * error`) throw (`exceptions::ActiveMQException`) [pure virtual]

6.202.3.3 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitor::processBrokerInfo` (`commands::BrokerInfo * info`) throw (`exceptions::ActiveMQException`) [pure virtual]

6.202.3.4 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitor::processCommitTransactionOnePhase`
(`commands::TransactionInfo * info`) throw (`exceptions::ActiveMQException`) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1296).

6.202.3.5 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitor::processCommitTransactionTwoPhase`
(`commands::TransactionInfo * info`) throw (`exceptions::ActiveMQException`) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1297).

6.202.3.6 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConnectionControl
(commands::ConnectionControl * *control*) throw (exceptions::ActiveMQException) [pure virtual]

6.202.3.7 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConnectionError
(commands::ConnectionError * *error*) throw (exceptions::ActiveMQException) [pure virtual]

6.202.3.8 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConnectionInfo
(commands::ConnectionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1297).

6.202.3.9 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConsumerControl
(commands::ConsumerControl * *control*) throw (exceptions::ActiveMQException) [pure virtual]

6.202.3.10 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConsumerInfo
(commands::ConsumerInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1297).

6.202.3.11 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processControlCommand
(commands::ControlCommand * *command*) throw (exceptions::ActiveMQException) [pure virtual]

6.202.3.12 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processDestinationInfo
(commands::DestinationInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1297).

6.202.3.13 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processEndTransaction
(commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1297).

- 6.202.3.14** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processFlushCommand
(commands::FlushCommand * command) throw (exceptions::ActiveMQException) [pure virtual]`
- 6.202.3.15** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processForgetTransaction
(commands::TransactionInfo * info) throw (exceptions::ActiveMQException) [pure virtual]`
- 6.202.3.16** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processKeepAliveInfo
(commands::KeepAliveInfo * info) throw (exceptions::ActiveMQException) [pure virtual]`
- 6.202.3.17** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessage (commands::Message * send) throw (exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1297).

- 6.202.3.18** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessageAck
(commands::MessageAck * ack) throw (exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1297).

- 6.202.3.19** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessageDispatch
(commands::MessageDispatch * dispatch) throw (exceptions::ActiveMQException) [pure virtual]`
- 6.202.3.20** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessageDispatchNotification
(commands::MessageDispatchNotification * notification) throw (exceptions::ActiveMQException) [pure virtual]`
- 6.202.3.21** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessagePull
(commands::MessagePull * pull) throw (exceptions::ActiveMQException) [pure virtual]`
- 6.202.3.22** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processPrepareTransaction
(commands::TransactionInfo * info) throw (exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1298).

6.202.3.23 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processProducerAck
(commands::ProducerAck * *ack*) throw (exceptions::ActiveMQException) [pure virtual]

6.202.3.24 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processProducerInfo
(commands::ProducerInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1298).

6.202.3.25 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRecoverTransactions
(commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

6.202.3.26 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveConnection (
commands::ConnectionId * *id*) throw (exceptions::ActiveMQException
) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1298).

6.202.3.27 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveConsumer (
commands::ConsumerId * *id*) throw (exceptions::ActiveMQException
) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1298).

6.202.3.28 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveDestination
(commands::DestinationInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1298).

6.202.3.29 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveInfo
(commands::RemoveInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::CommandVisitorAdapter` (p. 1123).

6.202.3.30 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveProducer (
commands::ProducerId * *id*) throw (exceptions::ActiveMQException
) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1298).

6.202.3.31 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveSession (
commands::SessionId * id) throw (exceptions::ActiveMQException)
[pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1298).

6.202.3.32 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveSubscriptionInfo
(commands::RemoveSubscriptionInfo * info) throw (
exceptions::ActiveMQException) [pure virtual]`

6.202.3.33 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processReplayCommand
(commands::ReplayCommand * replay) throw (
exceptions::ActiveMQException) [pure virtual]`

6.202.3.34 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processResponse
(commands::Response * response) throw (
exceptions::ActiveMQException) [pure virtual]`

6.202.3.35 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRollbackTransaction
(commands::TransactionInfo * info) throw (
exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1299).

6.202.3.36 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processSessionInfo
(commands::SessionInfo * info) throw (
exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1299).

6.202.3.37 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processShutdownInfo
(commands::ShutdownInfo * info) throw (
exceptions::ActiveMQException) [pure virtual]`

6.202.3.38 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processTransactionInfo
(commands::TransactionInfo * info) throw (
exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::CommandVisitorAdapter` (p. 1124).

```

6.202.3.39 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processWireFormat
( commands::WireFormatInfo * info ) throw (
exceptions::ActiveMQException ) [pure virtual]

```

The documentation for this class was generated from the following file:

- src/main/activemq/state/CommandVisitor.h

6.203 activemq::state::CommandVisitorAdapter Class Reference

Default Implementation of a **CommandVisitor** (p. 1113) that returns NULL for all calls.

```
#include <src/main/activemq/state/CommandVisitorAdapter.h>
```

Inheritance diagram for activemq::state::CommandVisitorAdapter:

Public Member Functions

- virtual **~CommandVisitorAdapter** ()
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConnection** (commands::ConnectionId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSession** (commands::SessionId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveProducer** (commands::ProducerId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConsumer** (commands::ConsumerId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processDestinationInfo** (commands::DestinationInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveDestination** (commands::DestinationInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo** (commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessage** (commands::Message *send AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageAck** (commands::MessageAck *ack AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessagePull** (commands::MessagePull *pull AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer**< **commands::Command** > **processBeginTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processPrepareTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processCommitTransactionOnePhase** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processCommitTransactionTwoPhase** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRollbackTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processFlushCommand** (**commands::FlushCommand** *command AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** *notification AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerAck** (**commands::ProducerAck** *ack AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** *dispatch AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processControlCommand** (**commands::ControlCommand** *command AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processConnectionError** (**commands::ConnectionError** *error AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConnectionControl** (**commands::ConnectionControl** *control AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConsumerControl** (**commands::ConsumerControl** *control AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processBrokerError** (**commands::BrokerError** *error AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processReplayCommand** (**commands::ReplayCommand** *replay AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processResponse** (**commands::Response** *response AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConnectionInfo** (**commands::ConnectionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processSessionInfo** (**commands::SessionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processProducerInfo** (**commands::ProducerInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConsumerInfo** (**commands::ConsumerInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processTransactionInfo** (**commands::TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveInfo** (**commands::RemoveInfo** *info) throw (exceptions::ActiveMQException)

6.203.1 Detailed Description

Default Implementation of a **CommandVisitor** (p.1113) that returns NULL for all calls.

Since

3.0

6.203.2 Constructor & Destructor Documentation

6.203.2.1 virtual
activemq::state::CommandVisitorAdapter::~~CommandVisitorAdapter () [inline, virtual]

6.203.3 Member Function Documentation

6.203.3.1 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBeginTransaction (commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.203.3.2 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBrokerError (commands::BrokerError *error *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.203.3.3 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBrokerInfo (commands::BrokerInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.203.3.4 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processCommitTransactionOnePhase (commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.203.3.5 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processCommitTransactionTwoPhase (commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.203.3.6 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionControl (commands::ConnectionControl *control *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.203.3.7 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionError (commands::ConnectionError *error *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.203.3.8 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionInfo (commands::ConnectionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.203.3.9 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConsumerControl (commands::ConsumerControl *control *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.203.3.10 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConsumerInfo (commands::ConsumerInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.203.3.11 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processControlCommand (commands::ControlCommand *command *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

References `activemq::commands::ConnectionId::ID_CONNECTIONID`, `activemq::commands::ConsumerId::ID_CONSUMERID`, `activemq::commands::ProducerId::ID_PRODUCERID`, and `activemq::commands::SessionId::ID_SESSIONID`.

- 6.203.3.30 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processRemoveProducer`
`(commands::ProducerId *id AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.31 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processRemoveSession`
`(commands::SessionId *id AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.32 `virtual decaf::lang::Pointer<commands::Command>` `ac-`
`tivemq::state::CommandVisitorAdapter::processRemoveSubscriptionInfo`
`(commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED)`
`throw (exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.33 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processReplayCommand`
`(commands::ReplayCommand *replay AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.34 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processResponse`
`(commands::Response *response AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.35 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processRollbackTransaction`
`(commands::TransactionInfo *info AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.36 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processSessionInfo`
`(commands::SessionInfo *info AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.37 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processShutdownInfo`
`(commands::ShutdownInfo *info AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.38 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processTransactionInfo`
`(commands::TransactionInfo * info) throw (`
`exceptions::ActiveMQException) [inline, virtual]`

Implements `activemq::state::CommandVisitor` (p. 1119).

References `activemq::core::ActiveMQConstants::TRANSACTION_STATE_BEGIN`,
`activemq::core::ActiveMQConstants::TRANSACTION_STATE_`

```

COMMITONEPHASE,      activemq::core::ActiveMQConstants::TRANSACTION_STATE_-
COMMITTWOPHASE,      activemq::core::ActiveMQConstants::TRANSACTION_-
STATE_END,           activemq::core::ActiveMQConstants::TRANSACTION_STATE_-
FORGET,              activemq::core::ActiveMQConstants::TRANSACTION_STATE_PREPARE,
activemq::core::ActiveMQConstants::TRANSACTION_STATE_RECOVER,      and
activemq::core::ActiveMQConstants::TRANSACTION_STATE_ROLLBACK.

```

```

6.203.3.39  virtual decaf::lang::Pointer<commands::Command>
               activemq::state::CommandVisitorAdapter::processWireFormat (
               commands::WireFormatInfo *info  AMQCPP_UNUSED ) throw (
               exceptions::ActiveMQException ) [inline, virtual]

```

The documentation for this class was generated from the following file:

- src/main/activemq/state/CommandVisitorAdapter.h

6.204 decaf::lang::Comparable< T > Class Template Reference

This interface imposes a total ordering on the objects of each class that implements it.

```
#include <src/main/decaf/lang/Comparable.h>
```

Public Member Functions

- virtual **~Comparable** ()
- virtual int **compareTo** (const T &value) const =0
Compares this object with the specified object for order.
- virtual bool **equals** (const T &value) const =0
- virtual bool **operator==** (const T &value) const =0
Compares equality between this object and the one passed.
- virtual bool **operator<** (const T &value) const =0
Compares this object to another and returns true if this object is considered to be less than the one passed.

6.204.1 Detailed Description

```
template<typename T> class decaf::lang::Comparable< T >
```

This interface imposes a total ordering on the objects of each class that implements it. This ordering is referred to as the class's natural ordering, and the class's compareTo method is referred to as its natural comparison method.

6.204.2 Constructor & Destructor Documentation

6.204.2.1 `template<typename T> virtual decaf::lang::Comparable< T >::~~Comparable () [inline, virtual]`

6.204.3 Member Function Documentation

6.204.3.1 `template<typename T> virtual int decaf::lang::Comparable< T >::compareTo (const T & value) const [pure virtual]`

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation `sgn(expression)` designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure `sgn(x.compareTo(y)) == -sgn(y.compareTo(x))` for all x and y. (This implies that `x.compareTo(y)` must throw an exception iff `y.compareTo(x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `(x.compareTo(y)>0 && y.compareTo(z)>0)` implies `x.compareTo(z)>0`.

Finally, the implementor must ensure that `x.compareTo(y)==0` implies that `sgn(x.compareTo(z)) == sgn(y.compareTo(z))`, for all z.

It is strongly recommended, but not strictly required that `(x.compareTo(y)==0) == (x.equals(y))`. Generally speaking, any class that implements the **Comparable** (p.1125) interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

Parameters

value - the Object to be compared.

Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Implemented in **decaf::lang::Boolean** (p.783), **decaf::lang::Byte** (p.887), **decaf::lang::Character** (p.1022), **decaf::lang::Double** (p.1675), **decaf::lang::Float** (p.1783), **decaf::lang::Integer** (p.1945), **decaf::lang::Long** (p.2271), and **decaf::lang::Short** (p.3223).

6.204.3.2 `template<typename T> virtual bool decaf::lang::Comparable< T >::equals (const T & value) const [pure virtual]`

Returns

true if this value is considered equal to the passed value.

Implemented in **decaf::lang::Boolean** (p.783), **decaf::lang::Byte** (p.888), **decaf::lang::Character** (p.1023), **decaf::lang::Double** (p.1677), **decaf::lang::Float** (p.1784), **decaf::lang::Integer** (p.1947), **decaf::lang::Long** (p.2273), and **decaf::lang::Short** (p.3224).

6.204.3.3 `template<typename T> virtual bool decaf::lang::Comparable< T >::operator< (const T & value) const` [pure virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implemented in `decaf::lang::Boolean` (p. 783), `decaf::lang::Byte` (p. 889), `decaf::lang::Character` (p. 1025), `decaf::lang::Double` (p. 1679), `decaf::lang::Float` (p. 1787), `decaf::lang::Integer` (p. 1949), `decaf::lang::Long` (p. 2275), and `decaf::lang::Short` (p. 3225).

6.204.3.4 `template<typename T> virtual bool decaf::lang::Comparable< T >::operator== (const T & value) const` [pure virtual]

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implemented in `decaf::lang::Boolean` (p. 784), `decaf::lang::Byte` (p. 890), `decaf::lang::Character` (p. 1025), `decaf::lang::Double` (p. 1680), `decaf::lang::Float` (p. 1788), `decaf::lang::Integer` (p. 1950), `decaf::lang::Long` (p. 2276), and `decaf::lang::Short` (p. 3226).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Comparable.h`

6.205 `decaf::util::Comparator< T >` Class Template Reference

A comparison function, which imposes a total ordering on some collection of objects.

```
#include <src/main/decaf/util/Comparator.h>
```

Public Member Functions

- `virtual ~Comparator()`

- virtual bool **operator()** (const T &left, const T &right) const =0

*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1127) to be passed to an STL **Map** (p. 2305) for use as the sorting criteria.*

- virtual int **compare** (const T &o1, const T &o2) const =0

Compares its two arguments for order.

6.205.1 Detailed Description

template<typename T> class decaf::util::Comparator< T >

A comparison function, which imposes a total ordering on some collection of objects. Comparators can be passed to a sort method (such as Collections.sort) to allow precise control over the sort order. Comparators can also be used to control the order of certain data structures.

The ordering imposed by a **Comparator** (p. 1127) c on a set of elements S is said to be consistent with equals if and only if (compare(e1, e2) == 0) has the same boolean value as (e1 == e2) for every e1 and e2 in S.

6.205.2 Constructor & Destructor Documentation

6.205.2.1 template<typename T> virtual decaf::util::Comparator< T >::~Comparator () [inline, virtual]

6.205.3 Member Function Documentation

6.205.3.1 template<typename T> virtual int decaf::util::Comparator< T >::compare (const T & o1, const T & o2) const [pure virtual]

Compares its two arguments for order.

Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that `sgn(compare(x, y)) == -sgn(compare(y, x))` for all x and y. (This implies that `compare(x, y)` must throw an exception if and only if `compare(y, x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `((compare(x, y)>0) && (compare(y, z)>0))` implies `compare(x, z)>0`.

Finally, the implementer must ensure that `compare(x, y)==0` implies that `sgn(compare(x, z))==sgn(compare(y, z))` for all z.

It is generally the case, but not strictly required that `(compare(x, y)==0) == (x == y)`. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

Parameters

o1 - the first object to be compared

o2 - the second object to be compared

Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implemented in **decaf::util::comparators::Less< E >** (p. 2183).

6.205.3.2 `template<typename T> virtual bool decaf::util::Comparator< T
>::operator() (const T & left, const T & right) const` [pure virtual]

Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1127) to be passed to an STL **Map** (p. 2305) for use as the sorting criteria.

Parameters

left - the Left hand side operand.

right - the Right hand side operand.

Returns

true if the vale of left is less than the value of right.

Implemented in **decaf::util::comparators::Less< E >** (p. 2183).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

6.206 activemq::util::CompositeData Class Reference

Represents a Composite URI.

```
#include <src/main/activemq/util/CompositeData.h>
```

Public Member Functions

- **CompositeData** ()
- virtual **~CompositeData** ()
- **StlList< URI > &getComponents** ()
- const **StlList< URI > &getComponents** () const
- void **setComponents** (const **StlList< URI > &components**)
- std::string **getFragment** () const
- void **setFragment** (const std::string &fragment)
- const **Properties &getParameters** () const
- void **setParameters** (const **Properties ¶meters**)
- std::string **getScheme** () const
- void **setScheme** (const std::string &scheme)
- std::string **getPath** () const
- void **setPath** (const std::string &path)
- std::string **getHost** () const
- void **setHost** (const std::string &host)
- **URI toURI** () const throw (decaf::net::URISyntaxException)

6.206.1 Detailed Description

Represents a Composite URI.

Since

3.0

6.206.2 Constructor & Destructor Documentation

6.206.2.1 `activemq::util::CompositeData::CompositeData ()`

6.206.2.2 `virtual activemq::util::CompositeData::~~CompositeData () [virtual]`

6.206.3 Member Function Documentation

6.206.3.1 `StlList<URI>& activemq::util::CompositeData::getComponents () [inline]`

6.206.3.2 `const StlList<URI>& activemq::util::CompositeData::getComponents () const [inline]`

6.206.3.3 `std::string activemq::util::CompositeData::getFragment () const [inline]`

6.206.3.4 `std::string activemq::util::CompositeData::getHost () const [inline]`

6.206.3.5 `const Properties& activemq::util::CompositeData::getParameters () const [inline]`

6.206.3.6 `std::string activemq::util::CompositeData::getPath () const [inline]`

6.206.3.7 `std::string activemq::util::CompositeData::getScheme () const [inline]`

6.206.3.8 `void activemq::util::CompositeData::setComponents (const StlList<URI> & components) [inline]`

6.206.3.9 `void activemq::util::CompositeData::setFragment (const std::string & fragment) [inline]`

6.206.3.10 `void activemq::util::CompositeData::setHost (const std::string & host) [inline]`

6.206.3.11 `void activemq::util::CompositeData::setParameters (const Properties & parameters) [inline]`

6.206.3.12 `void activemq::util::CompositeData::setPath (const std::string & path) [inline]`

6.206.3.13 `void activemq::util::CompositeData::setScheme (const std::string & scheme) [inline]`

6.206.3.14 `URI activemq::util::CompositeData::toURI () const throw (decaf::net::URISyntaxException)`

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CompositeData.h`

6.207 activemq::threads::CompositeTask Class Reference

Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p.1133).

```
#include <src/main/activemq/threads/CompositeTask.h>
```

Inheritance diagram for activemq::threads::CompositeTask:

Public Member Functions

- virtual `~CompositeTask()`
- virtual bool `isPending()` const =0

*Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.3494) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.*

6.207.1 Detailed Description

Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p.1133).

Since

3.0

6.207.2 Constructor & Destructor Documentation

6.207.2.1 virtual `activemq::threads::CompositeTask::~~CompositeTask()`
[inline, virtual]

6.207.3 Member Function Documentation

6.207.3.1 virtual bool `activemq::threads::CompositeTask::isPending()` const
[pure virtual]

Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.3494) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.

Since

3.0

Implemented in `activemq::transport::failover::BackupTransportPool` (p.694), `activemq::transport::failover::CloseTransportsTask` (p.1067), and `activemq::transport::failover::FailoverTransport` (p.1758).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/CompositeTask.h`

6.208 `activemq::threads::CompositeTaskRunner` Class Reference

A **Task** (p. 3494) Runner that can contain one or more `CompositeTasks` that are each checked for pending work and run if any is present in the order that the tasks were added.

```
#include <src/main/activemq/threads/CompositeTaskRunner.h>
```

Inheritance diagram for `activemq::threads::CompositeTaskRunner`:

Public Member Functions

- **CompositeTaskRunner** ()
- virtual **~CompositeTaskRunner** ()
- void **addTask** (`CompositeTask *task`)
*Adds a new **CompositeTask** (p. 1132) to the Set of Tasks that this class manages.*
- void **removeTask** (`CompositeTask *task`)
*Removes a **CompositeTask** (p. 1132) that was added previously.*
- virtual void **shutdown** (unsigned int timeout)
Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.
- virtual void **shutdown** ()
Shutdown once the task has finished and the TaskRunner's thread has exited.
- virtual void **wakeup** ()
*Signal the **TaskRunner** (p. 3496) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3494) instance will be run until its iterate method has returned false indicating it is done.*

Protected Member Functions

- virtual void **run** ()
Run method - called by the Thread class in the context of the thread.
- virtual bool **iterate** ()

6.208.1 Detailed Description

A **Task** (p. 3494) Runner that can contain one or more `CompositeTasks` that are each checked for pending work and run if any is present in the order that the tasks were added.

Since

3.0

6.208.2 Constructor & Destructor Documentation

6.208.2.1 `activemq::threads::CompositeTaskRunner::CompositeTaskRunner ()`

6.208.2.2 `virtual
activemq::threads::CompositeTaskRunner::~~CompositeTaskRunner ()
[virtual]`

6.208.3 Member Function Documentation

6.208.3.1 `void activemq::threads::CompositeTaskRunner::addTask (CompositeTask * task)`

Adds a new **CompositeTask** (p. 1132) to the Set of Tasks that this class manages.

Parameters

task - Pointer to a **CompositeTask** (p. 1132) instance.

6.208.3.2 `virtual bool activemq::threads::CompositeTaskRunner::iterate ()
[protected, virtual]`

6.208.3.3 `void activemq::threads::CompositeTaskRunner::removeTask (CompositeTask * task)`

Removes a **CompositeTask** (p. 1132) that was added previously.

Parameters

task - Pointer to a **CompositeTask** (p. 1132) instance.

6.208.3.4 `virtual void activemq::threads::CompositeTaskRunner::run ()
[protected, virtual]`

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3112).

6.208.3.5 `virtual void activemq::threads::CompositeTaskRunner::shutdown ()
[virtual]`

Shutdown once the task has finished and the TaskRunner's thread has exited.

6.208.3.6 `virtual void activemq::threads::CompositeTaskRunner::shutdown (unsigned int timeout) [virtual]`

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters

timeout - Time in Milliseconds to wait for the task to stop.

6.208.3.7 virtual void activemq::threads::CompositeTaskRunner::wakeup () [virtual]

Signal the **TaskRunner** (p. 3496) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3494) instance will be run until its iterate method has returned false indicating it is done.

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**CompositeTaskRunner.h**

6.209 activemq::transport::CompositeTransport Class Reference

A Composite **Transport** (p. 3629) is a **Transport** (p. 3629) implementation that is composed of several Transports.

```
#include <src/main/activemq/transport/CompositeTransport.h>
```

Inheritance diagram for activemq::transport::CompositeTransport:

Public Member Functions

- virtual ~**CompositeTransport** ()
- virtual void **addURI** (const **List**< **URI** > &uris)=0

*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3629) is a composite of.*

- virtual void **removeURI** (const **List**< **URI** > &uris)=0

*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3629) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3629) should result in that **Transport** (p. 3629) being disposed of.*

6.209.1 Detailed Description

A Composite **Transport** (p. 3629) is a **Transport** (p. 3629) implementation that is composed of several Transports. The composition could be such that only one **Transport** (p. 3629) exists for each URI that is composed or there could be many active Transports working at once.

Since

3.0

6.209.2 Constructor & Destructor Documentation

6.209.2.1 `virtual activemq::transport::CompositeTransport::~CompositeTransport () [inline, virtual]`

6.209.3 Member Function Documentation

6.209.3.1 `virtual void activemq::transport::CompositeTransport::addURI (const List< URI > & uris) [pure virtual]`

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3629) is a composite of.

Parameters

uris The new URI set to add to the set this composite maintains.

6.209.3.2 `virtual void activemq::transport::CompositeTransport::removeURI (const List< URI > & uris) [pure virtual]`

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3629) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3629) should result in that **Transport** (p. 3629) being disposed of.

Parameters

uris The new URI set to remove to the set this composite maintains.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/CompositeTransport.h`

6.210 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference

Interface for a **Map** (p. 2305) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2305) interface.

`#include <src/main/decaf/util/concurrent/ConcurrentMap.h>`

Inheritance diagram for decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >:

Public Member Functions

- `virtual ~ConcurrentMap ()`
- `virtual bool putIfAbsent (const K &key, const V &value)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)`

If the specified key is not already associated with a value, associate it with the given value.

- virtual bool **remove** (const K &key, const V &value)=0
Remove entry for key only if currently mapped to given value.
- virtual bool **replace** (const K &key, const V &oldValue, const V &newValue)=0
Replace entry for key only if currently mapped to given value.
- virtual V **replace** (const K &key, const V &value)=0 throw (decaf::lang::exceptions::NoSuchElementException)
Replace entry for key only if currently mapped to some value.

6.210.1 Detailed Description

template<typename K, typename V, typename COMPARATOR> class decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >

Interface for a **Map** (p. 2305) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2305) interface.

Since

1.0

6.210.2 Constructor & Destructor Documentation

6.210.2.1 template<typename K, typename V, typename COMPARATOR>
virtual decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR
>::~ConcurrentMap () [inline, virtual]

6.210.3 Member Function Documentation

6.210.3.1 template<typename K, typename V, typename COMPARATOR>
virtual bool decaf::util::concurrent::ConcurrentMap< K, V,
COMPARATOR >::putIfAbsent (const K & *key*, const V & *value*)
throw (decaf::lang::exceptions::UnsupportedOperationException) [pure
virtual]

If the specified key is not already associated with a value, associate it with the given value.

This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key The key to map the value to.

value The value to map to the given key.

Returns

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

Exceptions

UnsupportedOperationException if the put operation is not supported by this map

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p.1151), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p.1151), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p.1151), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p.1151), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p.1151), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p.1151), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p.1151).

6.210.3.2 template<typename K, typename V, typename COMPARATOR> virtual
bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR
>::remove (const K & *key*, const V & *value*) [pure virtual]

Remove entry for key only if currently mapped to given value.

Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == value ) ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.

value value associated with the specified key.

Returns

true if the value was removed, false otherwise

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p.1151), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >

(p. 1151), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1151), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1151), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1151), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1151), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1151).

6.210.3.3 `template<typename K, typename V, typename COMPARATOR> virtual bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::replace (const K & key, const V & oldValue, const V & newValue)` [pure virtual]

Replace entry for key only if currently mapped to given value.

Acts as

```
if ( ( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.

oldValue value expected to be associated with the specified key.

newValue value to be associated with the specified key.

Returns

true if the value was replaced

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1153), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1153), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1153), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1153), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1153), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1153), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1153).

6.210.3.4 `template<typename K, typename V, typename COMPARATOR> virtual
 V decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR
 >::replace (const K & key, const V & value) throw (
 decaf::lang::exceptions::NoSuchElementException) [pure virtual]`

Replace entry for key only if currently mapped to some value.

Acts as

```
if( ( map.containsKey( key ) ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.

value value to be associated with the specified key.

Returns

copy of the previous value associated with specified key, or throws an `NoSuchElementException` if there was no mapping for key.

Exceptions

NoSuchElementException if there was no previous mapping.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p.1152), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p.1152), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p.1152), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p.1152), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p.1152), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p.1152), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p.1152).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ConcurrentMap.h`

6.211 `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` Class Template Reference

Map (p.2305) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

```
#include <src/main/decaf/util/concurrent/ConcurrentStlMap.h>
```

Inheritance diagram for decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >:

Public Member Functions

- **ConcurrentStlMap** ()
Default constructor - does nothing.
- **ConcurrentStlMap** (const **ConcurrentStlMap** &source)
Copy constructor - copies the content of the given map into this one.
- **ConcurrentStlMap** (const **Map**< K, V, COMPARATOR > &source)
Copy constructor - copies the content of the given map into this one.
- virtual ~**ConcurrentStlMap** ()
- virtual bool **equals** (const **ConcurrentStlMap** &source) const
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const
Comparison, equality is dependent on the method of determining if the element are equal.
- virtual void **copy** (const **ConcurrentStlMap** &source)
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)
Copies the content of the source map into this map.
- virtual void **clear** () throw (decaf::lang::exceptions::UnsupportedOperationException)
Removes all keys and values from this map.
Exceptions
***UnsupportedOperationException** if this map is unmodifiable.*
- virtual bool **containsKey** (const K &key) const
Indicates whether or this map contains a value for the given key.
Parameters
***key** The key to look up.*
Returns
true if this map contains the value, otherwise false.
- virtual bool **containsValue** (const V &value) const
Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.
Parameters
***value** The Value to look up.*
Returns
true if this map contains the value, otherwise false.

- virtual bool **isEmpty** () const
Returns
*if the **Map** (p. 2305) contains any element or not, TRUE or FALSE*
- virtual std::size_t **size** () const
Returns
The number of elements (key/value pairs) in this map.
- virtual V & **get** (const K &key) throw (lang::exceptions::NoSuchElementException)
*Gets the value mapped to the specified key in the **Map** (p. 2305).*
*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*
Parameters
key The search key.
Returns
A reference to the value for the given key.
Exceptions
***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2305).*
- virtual const V & **get** (const K &key) const throw (lang::exceptions::NoSuchElementException)
*Gets the value mapped to the specified key in the **Map** (p. 2305).*
*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*
Parameters
key The search key.
Returns
A {const} reference to the value for the given key.
Exceptions
***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2305).*
- virtual void **put** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)
Sets the value for the specified key.
Parameters
key The target key.
value The value to be set.
Exceptions
***UnsupportedOperationException** if this map is unmodifiable.*
- virtual void **putAll** (const ConcurrentStlMap< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)
- virtual void **putAll** (const Map< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)
*Stores a copy of the Mappings contained in the other **Map** (p. 2305) in this one.*

Parameters

*other A **Map** (p. 2305) instance whose elements are to all be inserted in this **Map** (p. 2305).*

Exceptions

UnsupportedOperationException *If the implementing class does not support the putAll operation.*

- virtual V **remove** (const K &key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

key The search key.

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException *if this key is not in the **Map** (p. 2305).*
UnsupportedOperationException *if this map is unmodifiable.*

- virtual std::vector< K > **keySet** () const

*Returns a **Set** (p. 3220) view of the mappings contained in this map.*

*The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2014), **Set.remove** (p. 151), removeAll, retainAll and clear operations. It does not support the add or addAll operations.*

Returns

the entire set of keys in this map as a std::vector.

- virtual std::vector< V > **values** () const

Returns

the entire set of values in this map as a std::vector.

- bool **putIfAbsent** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)

If the specified key is not already associated with a value, associate it with the given value.

- bool **remove** (const K &key, const V &value)

Remove entry for key only if currently mapped to given value.

- bool **replace** (const K &key, const V &oldValue, const V &newValue)

Replace entry for key only if currently mapped to given value.

- V **replace** (const K &key, const V &value) throw (decaf::lang::exceptions::NoSuchElementException)

Replace entry for key only if currently mapped to some value.

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
*Attempts to **Lock** (p. 2228) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.211.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>>
 class decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >

Map (p. 2305) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map. This version of **Map** (p. 2305) extends the **ConcurrentMap** (p. 1136) interface and implements all the methods defined in that interface. Unlike a Java ConcurrentHashMap this implementations synchronizes all methods such that any call to this class will block if another thread is already holding a lock, much like the Java HashTable.

Since

1.0

6.211.2 Constructor & Destructor Documentation

6.211.2.1 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::ConcurrentStlMap () [inline]`

Default constructor - does nothing.

6.211.2.2 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::ConcurrentStlMap (const ConcurrentStlMap< K,
V, COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters

source The source map.

6.211.2.3 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K,
V, COMPARATOR >::ConcurrentStlMap (const Map< K, V,
COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters

source The source map.

6.211.2.4 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::~~ConcurrentStlMap () [inline, virtual]`

6.211.3 Member Function Documentation

6.211.3.1 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
K, V, COMPARATOR >::clear () throw (
decaf::lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Removes all keys and values from this map.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2307).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy()`.

6.211.3.2 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::containsKey (const K & key) const
 [inline, virtual]`

Indicates whether or this map contains a value for the given key.

Parameters

key The key to look up.

Returns

true if this map contains the value, otherwise false.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2308).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::remove()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace()`.

6.211.3.3 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::containsValue (const V & value) const
 [inline, virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters

value The Value to look up.

Returns

true if this map contains the value, otherwise false.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2309).

6.211.3.4 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::copy (const ConcurrentStlMap< K, V,
 COMPARATOR > & source) [inline, virtual]`

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::ConcurrentStlMap()`.

6.211.3.5 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::copy (const Map< K, V, COMPARATOR > & source) [inline, virtual]`

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters

source The source object to copy from.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2309).

6.211.3.6 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::equals (const Map< K, V, COMPARATOR > & source) const [inline, virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters

source - `Map` (p. 2305) to compare to this one.

Returns

true if the `Map` (p. 2305) passed is equal in value to this one.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2310).

6.211.3.7 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::equals (const ConcurrentStlMap< K, V, COMPARATOR > & source) const [inline, virtual]`

6.211.3.8 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V& decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::get (const K & key) throw (lang::exceptions::NoSuchElementException) [inline, virtual]`

Gets the value mapped to the specified key in the `Map` (p. 2305).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters

key The search key.

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 2305).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2310).

6.211.3.9 `template<typename K, typename V, typename
COMPARATOR = std::less<K>> virtual const V&
decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::get (const K & key) const throw (
lang::exceptions::NoSuchElementException) [inline, virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2305).

If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.

Parameters

key The search key.

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 2305).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2311).

6.211.3.10 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
K, V, COMPARATOR >::isEmpty () const [inline, virtual]`

Returns

if the **Map** (p. 2305) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2312).

6.211.3.11 `template<typename K, typename V, typename
COMPARATOR = std::less<K>> virtual std::vector<K>
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::keySet () const [inline, virtual]`

Returns a **Set** (p. 3220) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2014), **Set.remove** (p. 151), removeAll, retainAll and clear operations. It does not support the add or addAll operations.

Returns

the entire set of keys in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2313).

6.211.3.12 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3463).

6.211.3.13 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3464).

6.211.3.14 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3465).

6.211.3.15 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::put (const K & key, const V & value
) throw (decaf::lang::exceptions::UnsupportedOperationException)
 [inline, virtual]`

Sets the value for the specified key.

Parameters

key The target key.
value The value to be set.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2314).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace()`.

6.211.3.16 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::putAll (const Map<
 K, V, COMPARATOR > & other) throw (
 decaf::lang::exceptions::UnsupportedOperationException) [inline,
 virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 2305) in this one.

Parameters

other A **Map** (p. 2305) instance whose elements are to all be inserted in this **Map** (p. 2305).

Exceptions

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2314).

6.211.3.17 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::putAll (const ConcurrentStlMap<
 K, V, COMPARATOR > & other) throw (
 decaf::lang::exceptions::UnsupportedOperationException) [inline,
 virtual]`

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy()`.

6.211.3.18 `template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::putIfAbsent (const K & key, const V & value) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

If the specified key is not already associated with a value, associate it with the given value.

This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key The key to map the value to.

value The value to map to the given key.

Returns

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

Exceptions

UnsupportedOperationException if the put operation is not supported by this map

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p. 1137).

6.211.3.19 `template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::remove (const K & key, const V & value) [inline, virtual]`

Remove entry for key only if currently mapped to given value.

Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == value ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.
value value associated with the specified key.

Returns

true if the value was removed, false otherwise

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 1138).

6.211.3.20 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::remove (const K & key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

key The search key.

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException if this key is not in the **Map** (p. 2305).

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2315).

6.211.3.21 `template<typename K, typename V, typename COMPARATOR = std::less<K>> V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::replace (const K & key, const V & value) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Replace entry for key only if currently mapped to some value.

Acts as

```
if( map.containsKey( key ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.

value value to be associated with the specified key.

Returns

copy of the previous value associated with specified key, or throws an `NoSuchElementException` if there was no mapping for key.

Exceptions

NoSuchElementException if there was no previous mapping.

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p. 1140).

6.211.3.22 `template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::replace (const K & key, const V & oldValue, const V & newValue) [inline, virtual]`

Replace entry for key only if currently mapped to given value.

Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.

oldValue value expected to be associated with the specified key.

newValue value to be associated with the specified key.

Returns

true if the value was replaced

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p. 1139).

6.211.3.23 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::size_t decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::size () const [inline, virtual]`

Returns

The number of elements (key/value pairs) in this map.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2316).

6.211.3.24 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::tryLock () throw (
 decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to **Lock** (p. 2228) the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3466).

6.211.3.25 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::unlock () throw (
 decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3467).

6.211.3.26 `template<typename K, typename V, typename
 COMPARATOR = std::less<K>> virtual std::vector<V>
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
 >::values () const [inline, virtual]`

Returns

the entire set of values in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2317).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::values()`.

6.211.3.27 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::wait (long long millisecs
) throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException,
 decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3470).

```
6.211.3.28  template<typename K, typename V, typename
             COMPARATOR = std::less<K>> virtual void
             decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
             >::wait (    ) throw ( decaf::lang::exceptions::RuntimeException,
             decaf::lang::exceptions::IllegalMonitorStateException,
             decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3468).

```
6.211.3.29  template<typename K, typename V, typename COMPARATOR =
             std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
             K, V, COMPARATOR >::wait ( long long millisecs, int
             nanos ) throw ( decaf::lang::exceptions::RuntimeException,
             decaf::lang::exceptions::IllegalArgumentException,
             decaf::lang::exceptions::IllegalMonitorStateException,
             decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3471).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ConcurrentStlMap.h**

6.212 decaf::util::concurrent::locks::Condition Class Reference

Condition (p. 1156) factors out the **Mutex** (p. 2604) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2229) implementations.

```
#include <src/main/decaf/util/concurrent/locks/Condition.h>
```

Public Member Functions

- virtual **~Condition** ()
- virtual void **await** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)
Causes the current thread to wait until it is signaled or interrupted.
- virtual void **awaitUninterruptibly** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Causes the current thread to wait until it is signalled.
- virtual long long **awaitNanos** (long long nanosTimeout)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)
Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.
- virtual bool **await** (long long time, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

- virtual bool **awaitUntil** (const **Date** &deadline)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)
- virtual void **signal** ()=0 throw (decaf::lang::exceptions::RuntimeException)

Wakes up one waiting thread.

- virtual void **signalAll** ()=0 throw (decaf::lang::exceptions::RuntimeException)

Wakes up all waiting threads.

6.212.1 Detailed Description

Condition (p. 1156) factors out the **Mutex** (p. 2604) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2229) implementations. Where a **Lock** (p. 2229) replaces the use of synchronized statements, a **Condition** (p. 1156) replaces the use of the Object monitor methods.

Conditions (also known as condition queues or condition variables) provide a means for one thread to suspend execution (to "wait") until notified by another thread that some state condition may now be true. Because access to this shared state information occurs in different threads, it must be protected, so a lock of some form is associated with the condition. The key property that waiting for a condition provides is that it atomically releases the associated lock and suspends the current thread.

A **Condition** (p. 1156) instance is intrinsically bound to a lock. To obtain a **Condition** (p. 1156) instance for a particular **Lock** (p. 2229) instance use its newCondition() method.

As an example, suppose we have a bounded buffer which supports put and take methods. If a take is attempted on an empty buffer, then the thread will block until an item becomes available; if a put is attempted on a full buffer, then the thread will block until a space becomes available. We would like to keep waiting put threads and take threads in separate wait-sets so that we can use the optimization of only notifying a single thread at a time when items or spaces become available in the buffer. This can be achieved using two **Condition** (p. 1156) instances.

```
class BoundedBuffer { Lock* lock = new ReentrantLock(); Condition* notFull = lock->newCondition(); Condition* notEmpty = lock->newCondition();
```

```
Object* items = new Object[100]; int putptr, takeptr, count;
```

```
public void put( Object* x ) throw( InterruptedException ) { lock->lock(); try { while( count == 100 ) notFull->await() (p. 1158); items[putptr] = x; if (++putptr == 100) putptr = 0; ++count; notEmpty->signal() (p. 1162); } catch(...) { lock->unlock(); } }
```

```
public Object take() throw( InterruptedException ) { lock->lock(); try { while(count == 0) notEmpty->await() (p. 1158); Object x = items[takeptr]; if (++takeptr == 100) takeptr = 0; --count; notFull->signal() (p. 1162); return x; } catch(...) { lock->unlock(); } }
```

(The ArrayBlockingQueue class provides this functionality, so there is no reason to implement this sample usage class.)

Implementation Considerations

When waiting upon a **Condition** (p. 1156), a "spurious wakeup" is permitted to occur, in general, as a concession to the underlying platform semantics. This has little practical impact on most

application programs as a **Condition** (p.1156) should always be waited upon in a loop, testing the state predicate that is being waited for. An implementation is free to remove the possibility of spurious wakeups but it is recommended that applications programmers always assume that they can occur and so always wait in a loop.

The three forms of condition waiting (interruptible, non-interruptible, and timed) may differ in their ease of implementation on some platforms and in their performance characteristics. In particular, it may be difficult to provide these features and maintain specific semantics such as ordering guarantees. Further, the ability to interrupt the actual suspension of the thread may not always be feasible to implement on all platforms.

Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of waiting, nor is it required to support interruption of the actual suspension of the thread.

An implementation is required to clearly document the semantics and guarantees provided by each of the waiting methods, and when an implementation does support interruption of thread suspension then it must obey the interruption semantics as defined in this interface.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since

1.0

6.212.2 Constructor & Destructor Documentation

6.212.2.1 `virtual decaf::util::concurrent::locks::Condition::~~Condition ()`
[inline, virtual]

6.212.3 Member Function Documentation

6.212.3.1 `virtual void decaf::util::concurrent::locks::Condition::await () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)` [pure virtual]

Causes the current thread to wait until it is signaled or interrupted.

The lock associated with this **Condition** (p.1156) is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

* Some other thread invokes the **signal()** (p.1162) method for this **Condition** (p.1156) and the current thread happens to be chosen as the thread to be awakened; or * Some other thread invokes the **signalAll()** (p.1162) method for this **Condition** (p.1156); or * Some other thread interrupts the current thread, and interruption of thread suspension is supported; or * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting and interruption of thread suspension is supported,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p.1156) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal. In that case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p.1156).

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

```
6.212.3.2  virtual bool decaf::util::concurrent::locks::Condition::await
            ( long long time, const TimeUnit & unit )
            throw ( decaf::lang::exceptions::RuntimeException,
                    decaf::lang::exceptions::InterruptedException,
                    decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]
```

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

This method is behaviorally equivalent to:

```
awaitNanos(unit.toNanos(time)) > 0
```

Parameters

time - the maximum time to wait

unit - the time unit of the time argument

Returns

false if the waiting time detectably elapsed before return from the method, else true

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p.1156).

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

6.212.3.3 `virtual long long decaf::util::concurrent::locks::Condition::awaitNanos(long long nanosTimeout) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]`

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of five things happens:

- * Some other thread invokes the **signal()** (p. 1162) method for this **Condition** (p. 1156) and the current thread happens to be chosen as the thread to be awakened; or
- * Some other thread invokes the **signalAll()** (p. 1162) method for this **Condition** (p. 1156); or
- * Some other thread interrupts the current thread, and interruption of thread suspension is supported; or
- * The specified waiting time elapses; or
- * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

- * has its interrupted status set on entry to this method; or
- * is interrupted while waiting and interruption of thread suspension is supported,

then **InterruptedException** is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

The method returns an estimate of the number of nanoseconds remaining to wait given the supplied *nanosTimeout* value upon return, or a value less than or equal to zero if it timed out. This value can be used to determine whether and how long to re-wait in cases where the wait returns but an awaited condition still does not hold. Typical uses of this method take the following form:

```
synchronized boolean aMethod( long timeout, const TimeUnit& unit ) { long nanosTimeout = unit.toNanos(timeout); while (!conditionBeingWaitedFor) { if (nanosTimeout > 0) nanosTimeout = theCondition->awaitNanos(nanosTimeout); else return false; } // ... }
```

Design note: This method requires a nanosecond argument so as to avoid truncation errors in reporting remaining times. Such precision loss would make it difficult for programmers to ensure that total waiting times are not systematically shorter than specified when re-waits occur.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1156) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as **IllegalMonitorStateException**) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal, or over indicating the elapse of the specified waiting time. In either case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

Parameters

nanosTimeout - the maximum time to wait, in nanoseconds

Returns

an estimate of the `nanosTimeout` value minus the time spent waiting upon return from this method. A positive value may be used as the argument to a subsequent call to this method to finish waiting out the desired time. A value less than or equal to zero indicates that no time remains.

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p. 1156).

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

6.212.3.4 `virtual void decaf::util::concurrent::locks::Condition::awaitUninterruptibly () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]`

Causes the current thread to wait until it is signalled.

The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* Some other thread invokes the **signal()** (p. 1162) method for this **Condition** (p. 1156) and the current thread happens to be chosen as the thread to be awakened; or * Some other thread invokes the **signalAll()** (p. 1162) method for this **Condition** (p. 1156); or * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread's interrupted status is set when it enters this method, or it is interrupted while waiting, it will continue to wait until signalled. When it finally returns from this method its interrupted status will still be set.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1156) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p. 1156).

IllegalMonitorStateException if the caller is not the lock owner.

6.212.3.5 virtual bool decaf::util::concurrent::locks::Condition::awaitUntil (const Date & *deadline*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

6.212.3.6 virtual void decaf::util::concurrent::locks::Condition::signal () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Wakes up one waiting thread.

If any threads are waiting on this condition then one is selected for waking up. That thread must then re-acquire the lock before returning from await.

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p.1156).

6.212.3.7 virtual void decaf::util::concurrent::locks::Condition::signalAll () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Wakes up all waiting threads.

If any threads are waiting on this condition then they are all woken up. Each thread must re-acquire the lock before it can return from await.

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p.1156).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**Condition.h**

6.213 decaf::util::concurrent::ConditionHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h>
```

Public Member Functions

- ConditionHandle ()
- ~ConditionHandle ()
- ConditionHandle ()
- ~ConditionHandle ()

Data Fields

- `pthread_cond_t condition`
- `MutexHandle * mutex`
- `HANDLE semaphore`
- `CRITICAL_SECTION criticalSection`
- volatile unsigned int `numWaiting`
- volatile unsigned int `numWake`
- volatile unsigned int `generation`

6.213.1 Constructor & Destructor Documentation

- 6.213.1.1 `decaf::util::concurrent::ConditionHandle ()`
[inline]
- 6.213.1.2 `decaf::util::concurrent::ConditionHandle::~~ConditionHandle ()`
[inline]
- 6.213.1.3 `decaf::util::concurrent::ConditionHandle::ConditionHandle ()`
[inline]
- 6.213.1.4 `decaf::util::concurrent::ConditionHandle::~~ConditionHandle ()`
[inline]

6.213.2 Field Documentation

- 6.213.2.1 `pthread_cond_t decaf::util::concurrent::ConditionHandle::condition`
- 6.213.2.2 `CRITICAL_SECTION decaf::util::concurrent::ConditionHandle::criticalSection`
- 6.213.2.3 `volatile unsigned int decaf::util::concurrent::ConditionHandle::generation`
- 6.213.2.4 `MutexHandle * decaf::util::concurrent::ConditionHandle::mutex`
- 6.213.2.5 `volatile unsigned int decaf::util::concurrent::ConditionHandle::numWaiting`
- 6.213.2.6 `volatile unsigned int decaf::util::concurrent::ConditionHandle::numWake`
- 6.213.2.7 `HANDLE decaf::util::concurrent::ConditionHandle::semaphore`

The documentation for this class was generated from the following files:

- `src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h`
- `src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h`

6.214 `decaf::internal::util::concurrent::ConditionImpl` Class Reference

```
#include <src/main/decaf/internal/util/concurrent/ConditionImpl.h>
```

Static Public Member Functions

- static **decaf::util::concurrent::ConditionHandle** * **create** (**decaf::util::concurrent::MutexHandle** *mutex)
Creates the Condition object and attaches it to the given MutexHandle.
- static void **destroy** (**decaf::util::concurrent::ConditionHandle** *handle)
Destroy a previously create Condition instance.
- static void **wait** (**decaf::util::concurrent::ConditionHandle** *condition)
Waits for the condition to be signaled.
- static void **wait** (**decaf::util::concurrent::ConditionHandle** *condition, long long mills, long long nanos)
Waits for the condition to be signaled or for the time specified to ellapse.
- static void **notify** (**decaf::util::concurrent::ConditionHandle** *condition)
Signals one Thread that is waiting on this condition to wake up.
- static void **notifyAll** (**decaf::util::concurrent::ConditionHandle** *condition)
Signals all Threads that is waiting on this condition to wake up.

6.214.1 Member Function Documentation

6.214.1.1 static **decaf::util::concurrent::ConditionHandle***
decaf::internal::util::concurrent::ConditionImpl::create (
decaf::util::concurrent::MutexHandle * *mutex*) [static]

Creates the Condition object and attaches it to the given MutexHandle.

Parameters

mutex the Mutex handle that this Condition is attached to.

Returns

a newly constructed Condition handle that is attached to the given handle.

6.214.1.2 static void **decaf::internal::util::concurrent::ConditionImpl::destroy** (
decaf::util::concurrent::ConditionHandle * *handle*) [static]

Destroy a previously create Condition instance.

Parameters

handle The Condition handle to be destroyed.

6.214.1.3 `static void decaf::internal::util::concurrent::ConditionImpl::notify (decaf::util::concurrent::ConditionHandle * condition) [static]`

Signals one Thread that is waiting on this condition to wake up.

Parameters

condition the handle to the condition to wait on.

6.214.1.4 `static void decaf::internal::util::concurrent::ConditionImpl::notifyAll (decaf::util::concurrent::ConditionHandle * condition) [static]`

Signals all Threads that is waiting on this condition to wake up.

Parameters

condition the handle to the condition to wait on.

6.214.1.5 `static void decaf::internal::util::concurrent::ConditionImpl::wait (decaf::util::concurrent::ConditionHandle * condition, long long mills, long long nanos) [static]`

Waits for the condition to be signaled or for the time specified to elapse.

Parameters

condition the handle to the condition to wait on.

mills the time in milliseconds to wait for the condition to be signaled.

nanos additional time in nanoseconds to wait for the thread to be signaled.

6.214.1.6 `static void decaf::internal::util::concurrent::ConditionImpl::wait (decaf::util::concurrent::ConditionHandle * condition) [static]`

Waits for the condition to be signaled.

Parameters

condition the handle to the condition to wait on.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/ConditionImpl.h`

6.215 decaf::net::ConnectException Class Reference

```
#include <src/main/decaf/net/ConnectException.h>
```

Inheritance diagram for decaf::net::ConnectException:

Public Member Functions

- **ConnectException** () throw ()
Default Constructor.
- **ConnectException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **ConnectException** (const **ConnectException** &ex) throw ()
Copy Constructor.
- **ConnectException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ConnectException** (const std::exception *cause) throw ()
Constructor.
- **ConnectException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ConnectException** * **clone** () const
Clones this exception.
- virtual ~**ConnectException** () throw ()

6.215.1 Constructor & Destructor Documentation

6.215.1.1 decaf::net::ConnectException::ConnectException () throw () [inline]

Default Constructor.

6.215.1.2 decaf::net::ConnectException::ConnectException (const Exception &ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.215.1.3 decaf::net::ConnectException::ConnectException (const ConnectException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.215.1.4 `decaf::net::ConnectException::ConnectException (const char * file,
const int lineNumber, const std::exception * cause, const char * msg,
...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.215.1.5 `decaf::net::ConnectException::ConnectException (const std::exception *
cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.215.1.6 `decaf::net::ConnectException::ConnectException (const char * file,
const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.215.1.7 `virtual decaf::net::ConnectException::~~ConnectException () throw ()
[inline, virtual]`

6.215.2 Member Function Documentation

6.215.2.1 `virtual ConnectException* decaf::net::ConnectException::clone ()
const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3300).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ConnectException.h**

6.216 cms::Connection Class Reference

The client's connection to its provider.

```
#include <src/main/cms/Connection.h>
```

Inheritance diagram for cms::Connection:

Public Member Functions

- virtual **~Connection** ()
- virtual void **close** ()=0 throw (CMSEException)
Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).
- virtual const **ConnectionMetaData** * **getMetaData** () const =0 throw (CMSEException)
Gets the metadata for this connection.
- virtual **Session** * **createSession** ()=0 throw (CMSEException)
Creates an AUTO_ACKNOWLEDGE Session (p. 3148).
- virtual **Session** * **createSession** (Session::AcknowledgeMode ackMode)=0 throw (CMSEException)
Creates a new Session (p. 3148) to work for this Connection (p. 1168) using the specified acknowledgment mode.
- virtual std::string **getClientID** () const =0
Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.
- virtual void **setClientID** (const std::string &clientID)=0
Sets the client identifier for this connection.
- virtual **ExceptionListener** * **getExceptionListener** () const =0
Gets the registered Exception Listener for this connection.
- virtual void **setExceptionListener** (ExceptionListener *listener)=0
Sets the registered Exception Listener for this connection.

6.216.1 Detailed Description

The client's connection to its provider. Connections support concurrent use.

A connection serves several purposes:

- It encapsulates an open connection with a JMS provider. It typically represents an open TCP/IP socket between a client and the service provider software.
- Its creation is where client authentication takes place.
- It can specify a unique client identifier.
- It provides a **ConnectionMetaData** (p. 1287) object.
- It supports an optional **ExceptionListener** (p. 1719) object.

Because the creation of a connection involves setting up authentication and communication, a connection is a relatively heavy-weight object. Most clients will do all their messaging with a single connection. Other more advanced applications may use several connections. The CMS API does not architect a reason for using multiple connections; however, there may be operational reasons for doing so.

A CMS client typically creates a connection, one or more sessions, and a number of message producers and consumers. When a connection is created, it is in stopped mode. That means that no messages are being delivered.

It is typical to leave the connection in stopped mode until setup is complete (that is, until all message consumers have been created). At that point, the client calls the connection's start method, and messages begin arriving at the connection's consumers. This setup convention minimizes any client confusion that may result from asynchronous message delivery while the client is still in the process of setting itself up.

A connection can be started immediately, and the setup can be done afterwards. Clients that do this must be prepared to handle asynchronous message delivery while they are still in the process of setting up.

A message producer can send messages while a connection is stopped.

Since

1.0

6.216.2 Constructor & Destructor Documentation

6.216.2.1 `virtual cms::Connection::~~Connection () [inline, virtual]`

6.216.3 Member Function Documentation

6.216.3.1 `virtual void cms::Connection::close () throw (CMSException) [pure virtual]`

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions

CMSException (p. 1074)

Implements **cms::Closeable** (p.1065).

6.216.3.2 `virtual Session* cms::Connection::createSession (Session::AcknowledgeMode ackMode) throw (CMSEException)` [pure virtual]

Creates a new **Session** (p.3148) to work for this **Connection** (p.1168) using the specified acknowledgment mode.

Parameters

ackMode the Acknowledgment Mode to use.

Exceptions

CMSEException (p. 1074)

6.216.3.3 `virtual Session* cms::Connection::createSession () throw (CMSEException)` [pure virtual]

Creates an AUTO_ACKNOWLEDGE **Session** (p.3148).

Exceptions

CMSEException (p. 1074)

6.216.3.4 `virtual std::string cms::Connection::getClientID () const` [pure virtual]

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.

Returns

Client Id String for this **Connection** (p.1168).

Exceptions

CMSEException (p. 1074) if the provider fails to return the client id or an internal error occurs.

6.216.3.5 `virtual ExceptionListener* cms::Connection::getExceptionListener () const` [pure virtual]

Gets the registered Exception Listener for this connection.

Returns

pointer to an exception listener or NULL

6.216.3.6 `virtual const ConnectionMetaData* cms::Connection::getMetaData ()
const throw (CMSEException) [pure virtual]`

Gets the metadata for this connection.

Returns

the connection MetaData pointer (caller does not own it).

Exceptions

CMSEException (p. 1074) if the provider fails to get the connection metadata for this connection.

See also

ConnectionMetaData (p. 1287)

Since

2.0

6.216.3.7 `virtual void cms::Connection::setClientID (const std::string & clientID
) [pure virtual]`

Sets the client identifier for this connection.

The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 1228) object and transparently assigned to the **Connection** (p. 1168) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 1868).

Parameters

clientID The unique client identifier to assign to the **Connection** (p. 1168).

Exceptions

CMSEException (p. 1074) if the provider fails to set the client id due to some internal error.

InvalidClientIDException if the id given is somehow invalid or is a duplicate.

IllegalStateException (p. 1868) if the client tries to set the id after a **Connection** (p. 1168) method has been called.

6.216.3.8 `virtual void cms::Connection::setExceptionListener (ExceptionListener
* listener) [pure virtual]`

Sets the registered Exception Listener for this connection.

Parameters

listener pointer to and **ExceptionListener** (p. 1719)

The documentation for this class was generated from the following file:

- `src/main/cms/Connection.h`

6.217 activemq::commands::ConnectionControl Class Reference

```
#include <src/main/activemq/commands/ConnectionControl.h>
```

Inheritance diagram for activemq::commands::ConnectionControl:

Public Member Functions

- **ConnectionControl** ()
- virtual **~ConnectionControl** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConnectionControl** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual bool **isExit** () const
- virtual void **setExit** (bool exit)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool faultTolerant)
- virtual bool **isResume** () const
- virtual void **setResume** (bool resume)
- virtual bool **isSuspend** () const
- virtual void **setSuspend** (bool suspend)
- virtual const std::string & **getConnectedBrokers** () const
- virtual std::string & **getConnectedBrokers** ()
- virtual void **setConnectedBrokers** (const std::string &connectedBrokers)
- virtual const std::string & **getReconnectTo** () const
- virtual std::string & **getReconnectTo** ()
- virtual void **setReconnectTo** (const std::string &reconnectTo)
- virtual bool **isRebalanceConnection** () const
- virtual void **setRebalanceConnection** (bool rebalanceConnection)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONCONTROL** = 18

Protected Attributes

- bool **close**
- bool **exit**
- bool **faultTolerant**
- bool **resume**
- bool **suspend**
- std::string **connectedBrokers**
- std::string **reconnectTo**
- bool **rebalanceConnection**

6.217.1 Constructor & Destructor Documentation

6.217.1.1 `activemq::commands::ConnectionControl::ConnectionControl ()`

6.217.1.2 `virtual activemq::commands::ConnectionControl::~~ConnectionControl () [virtual]`

6.217.2 Member Function Documentation

6.217.2.1 `virtual ConnectionControl* activemq::commands::ConnectionControl::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.217.2.2 `virtual void activemq::commands::ConnectionControl::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.217.2.3 `virtual bool activemq::commands::ConnectionControl::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.217.2.4 `virtual const std::string& activemq::commands::ConnectionControl::getConnectedBrokers () const` [virtual]

6.217.2.5 `virtual std::string& activemq::commands::ConnectionControl::getConnectedBrokers ()` [virtual]

6.217.2.6 `virtual unsigned char activemq::commands::ConnectionControl::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.217.2.7 `virtual const std::string& activemq::commands::ConnectionControl::getReconnectTo () const [virtual]`
- 6.217.2.8 `virtual std::string& activemq::commands::ConnectionControl::getReconnectTo () [virtual]`
- 6.217.2.9 `virtual bool activemq::commands::ConnectionControl::isClose () const [virtual]`
- 6.217.2.10 `virtual bool activemq::commands::ConnectionControl::isExit () const [virtual]`
- 6.217.2.11 `virtual bool activemq::commands::ConnectionControl::isFaultTolerant () const [virtual]`
- 6.217.2.12 `virtual bool activemq::commands::ConnectionControl::isRebalanceConnection () const [virtual]`
- 6.217.2.13 `virtual bool activemq::commands::ConnectionControl::isResume () const [virtual]`
- 6.217.2.14 `virtual bool activemq::commands::ConnectionControl::isSuspend () const [virtual]`
- 6.217.2.15 `virtual void activemq::commands::ConnectionControl::setClose (bool close) [virtual]`
- 6.217.2.16 `virtual void activemq::commands::ConnectionControl::setConnectedBrokers (const std::string & connectedBrokers) [virtual]`
- 6.217.2.17 `virtual void activemq::commands::ConnectionControl::setExit (bool exit) [virtual]`
- 6.217.2.18 `virtual void activemq::commands::ConnectionControl::setFaultTolerant (bool faultTolerant) [virtual]`
- 6.217.2.19 `virtual void activemq::commands::ConnectionControl::setRebalanceConnection (bool rebalanceConnection) [virtual]`
- 6.217.2.20 `virtual void activemq::commands::ConnectionControl::setReconnectTo (const std::string & reconnectTo) [virtual]`
- 6.217.2.21 `virtual void activemq::commands::ConnectionControl::setResume (bool resume) [virtual]`
- 6.217.2.22 `virtual void activemq::commands::ConnectionControl::setSuspend (bool suspend) [virtual]`
- 6.217.2.23 `virtual std::string activemq::commands::ConnectionControl::toString () const [virtual]`

Generated on Sun Sep 11 2011 18:23:35 for activemq-cpp-3.2.1 by Doxygen

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

6.217.2.24 `virtual Pointer<Command> activemq::commands::ConnectionControl::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.217.3 Field Documentation

- 6.217.3.1** `bool activemq::commands::ConnectionControl::close` [protected]
- 6.217.3.2** `std::string activemq::commands::ConnectionControl::connectedBrokers` [protected]
- 6.217.3.3** `bool activemq::commands::ConnectionControl::exit` [protected]
- 6.217.3.4** `bool activemq::commands::ConnectionControl::faultTolerant` [protected]
- 6.217.3.5** `const unsigned char activemq::commands::ConnectionControl::ID _- CONNECTIONCONTROL = 18` [static]
- 6.217.3.6** `bool activemq::commands::ConnectionControl::rebalanceConnection` [protected]
- 6.217.3.7** `std::string activemq::commands::ConnectionControl::reconnectTo` [protected]
- 6.217.3.8** `bool activemq::commands::ConnectionControl::resume` [protected]
- 6.217.3.9** `bool activemq::commands::ConnectionControl::suspend` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionControl.h`

6.218 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1177).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.218.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1177). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.218.2 Constructor & Destructor Documentation

6.218.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::ConnectionControlMarshaller () [inline]`

6.218.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::~~ConnectionControlMarshaller () [inline, virtual]`

6.218.3 Member Function Documentation

6.218.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.218.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.218.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.218.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.218.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

6.218.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

6.218.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h`

6.219 `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionControlMarshaller` (p. 1180).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller`:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.219.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p.1180). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.219.2 Constructor & Destructor Documentation

6.219.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::ConnectionControlMarshaller () [inline]`

6.219.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::~~ConnectionControlMarshaller () [inline, virtual]`

6.219.3 Member Function Documentation

6.219.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.219.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.219.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.219.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

6.219.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

6.219.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

6.219.3.7 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightUnmarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure`
`* dataStructure, decaf::io::DataInputStream * dataIn,`
`utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h`

6.220 `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionControlMarshaller` (p. 1184).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h>`

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller`:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.220.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p.1184). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.220.2 Constructor & Destructor Documentation

6.220.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::ConnectionControlMarshaller () [inline]`

6.220.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::~~ConnectionControlMarshaller () [inline, virtual]`

6.220.3 Member Function Documentation

6.220.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.220.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.220.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.220.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.220.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal1(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

6.220.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 719).

6.220.3.7 virtual void ac-
 tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h

6.221 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1188).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.221.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p.1188). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.221.2 Constructor & Destructor Documentation

6.221.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::ConnectionControlMarshaller () [inline]`

6.221.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::~~ConnectionControlMarshaller () [inline, virtual]`

6.221.3 Member Function Documentation

6.221.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.221.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.221.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.221.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

6.221.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal1(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

6.221.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 726).

6.221.3.7 virtual void ac-
 tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h

6.222 activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1192).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.222.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p.1192). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.222.2 Constructor & Destructor Documentation

6.222.2.1 `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::ConnectionControlMarshaller () [inline]`

6.222.2.2 `virtual
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::~~ConnectionControlMarshaller () [inline, virtual]`

6.222.3 Member Function Documentation

6.222.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.222.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.222.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.222.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

6.222.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightMarshal1(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

6.222.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 732).

6.222.3.7 virtual void ac-
 tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h

6.223 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1196).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.223.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p.1196). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.223.2 Constructor & Destructor Documentation

6.223.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::ConnectionControlMarshaller()` `[inline]`

6.223.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` `[inline, virtual]`

6.223.3 Member Function Documentation

6.223.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::createObject() const` `[virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.223.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.223.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.223.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

6.223.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal1(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

6.223.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 739).

```
6.223.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h

6.224 activemq::commands::ConnectionError Class Reference

```
#include <src/main/activemq/commands/ConnectionError.h>
```

Inheritance diagram for activemq::commands::ConnectionError:

Public Member Functions

- **ConnectionError** ()
- virtual **~ConnectionError** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConnectionError * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONERROR** = 16

Protected Attributes

- **Pointer**< **BrokerError** > exception
- **Pointer**< **ConnectionId** > connectionId

6.224.1 Constructor & Destructor Documentation

6.224.1.1 `activemq::commands::ConnectionError::ConnectionError ()`

6.224.1.2 `virtual activemq::commands::ConnectionError::~~ConnectionError ()`
[virtual]

6.224.2 Member Function Documentation

6.224.2.1 `virtual ConnectionError* activemq::commands::ConnectionError::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.224.2.2 `virtual void activemq::commands::ConnectionError::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.224.2.3 `virtual bool activemq::commands::ConnectionError::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

- 6.224.2.4 `virtual const Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId () const [virtual]`
- 6.224.2.5 `virtual Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId () [virtual]`
- 6.224.2.6 `virtual unsigned char activemq::commands::ConnectionError::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.224.2.7 `virtual Pointer<BrokerError>& activemq::commands::ConnectionError::getException () [virtual]`
- 6.224.2.8 `virtual const Pointer<BrokerError>& activemq::commands::ConnectionError::getException () const [virtual]`
- 6.224.2.9 `virtual void activemq::commands::ConnectionError::setConnectionId (const Pointer< ConnectionId > & connectionId) [virtual]`
- 6.224.2.10 `virtual void activemq::commands::ConnectionError::setException (const Pointer< BrokerError > & exception) [virtual]`
- 6.224.2.11 `virtual std::string activemq::commands::ConnectionError::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

- 6.224.2.12 `virtual Pointer<Command> activemq::commands::ConnectionError::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.224.3 Field Documentation

6.224.3.1 `Pointer<ConnectionId> activemq::commands::ConnectionError::connectionId`
[protected]

6.224.3.2 `Pointer<BrokerError> activemq::commands::ConnectionError::exception`
[protected]

6.224.3.3 `const unsigned char activemq::commands::ConnectionError::ID - CONNECTIONERROR = 16` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionError.h`

6.225 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1204).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller`:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.225.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1204). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.225.2 Constructor & Destructor Documentation

6.225.2.1 **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::ConnectionErrorMarshaller** () [inline]

6.225.2.2 virtual **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller** () [inline, virtual]

6.225.3 Member Function Documentation

6.225.3.1 virtual **commands::DataStructure*** **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.225.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::getDataStructureType() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.225.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 736).

6.225.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseUnmarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 737).

6.225.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

6.225.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

6.225.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h`

6.226 `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMa` Class Reference

Marshaling code for Open Wire Format for `ConnectionErrorMarshaller` (p. 1208).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller`:

Public Member Functions

- `ConnectionErrorMarshaller ()`
- `virtual ~ConnectionErrorMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.226.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1208). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.226.2 Constructor & Destructor Documentation

6.226.2.1 **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::ConnectionErrorMarshaller** () [inline]

6.226.2.2 **virtual**
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller () [inline, virtual]

6.226.3 Member Function Documentation

6.226.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.226.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.226.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.226.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.226.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

```
6.226.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.226.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h

6.227 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1212).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.227.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1212). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.227.2 Constructor & Destructor Documentation

6.227.2.1 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::ConnectionErrorMarshaller () [inline]

6.227.2.2 virtual activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller () [inline, virtual]

6.227.3 Member Function Documentation

6.227.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.227.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.227.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 709).

6.227.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 710).

6.227.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

```
6.227.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.227.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h`

6.228 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1216).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller**:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.228.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1216). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.228.2 Constructor & Destructor Documentation

6.228.2.1 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::ConnectionErrorMarshaller () [inline]

6.228.2.2 virtual activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller () [inline, virtual]

6.228.3 Member Function Documentation

6.228.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.228.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.228.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.228.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.228.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

```
6.228.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

```
6.228.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionErrorMarshaller.h**

6.229 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1220).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.229.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1220). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.229.2 Constructor & Destructor Documentation

6.229.2.1 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::ConnectionErrorMarshaller () [inline]

6.229.2.2 virtual activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller () [inline, virtual]

6.229.3 Member Function Documentation

6.229.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.229.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.229.3.3 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 722).

6.229.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 723).

6.229.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```
6.229.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.229.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h`

6.230 `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMa` Class Reference

Marshaling code for Open Wire Format for `ConnectionErrorMarshaller` (p. 1224).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller`:

Public Member Functions

- `ConnectionErrorMarshaller ()`
- `virtual ~ConnectionErrorMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.230.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1224). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.230.2 Constructor & Destructor Documentation

6.230.2.1 `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::ConnectionErrorMarshaller () [inline]`

6.230.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::~ConnectionErrorMarshaller () [inline, virtual]`

6.230.3 Member Function Documentation

6.230.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.230.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).


```

6.230.3.3  virtual void ac-
            tivemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::looseMarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
              decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

```

6.230.3.4  virtual void ac-
            tivemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::looseUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn ) throw (
              decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

```

6.230.3.5  virtual int ac-
            tivemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure
              * dataStructure, utils::BooleanStream * bs ) throw (
              decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

```
6.230.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

```
6.230.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionErrorMarshaller.h**

6.231 cms::ConnectionFactory Class Reference

Defines the interface for a factory that creates connection objects, the **Connection** (p.1168) objects returned implement the CMS **Connection** (p.1168) interface and hide the CMS Provider specific implementation details behind that interface.

```
#include <src/main/cms/ConnectionFactory.h>
```

Inheritance diagram for cms::ConnectionFactory:

Public Member Functions

- virtual **~ConnectionFactory** ()
- virtual **Connection * createConnection** ()=0 throw (CMSEException)
Creates a connection with the default user identity.
- virtual **cms::Connection * createConnection** (const std::string &username, const std::string &password)=0 throw (cms::CMSEException)
Creates a connection with the default specified identity.
- virtual **cms::Connection * createConnection** (const std::string &username, const std::string &password, const std::string &clientId)=0 throw (cms::CMSEException)
Creates a connection with the specified user identity.

Static Public Member Functions

- static **ConnectionFactory * createCMSConnectionFactory** (const std::string &brokerURI) throw (cms::CMSEException)
Static method that is used to create a provider specific connection factory.

6.231.1 Detailed Description

Defines the interface for a factory that creates connection objects, the **Connection** (p.1168) objects returned implement the CMS **Connection** (p.1168) interface and hide the CMS Provider specific implementation details behind that interface. A Client creates a new **ConnectionFactory** (p.1228) either directly by instantiating the provider specific implementation of the factory or

by using the static method `createCMSConnectionFactory` which all providers are required to implement.

Since

1.0

6.231.2 Constructor & Destructor Documentation

6.231.2.1 `virtual cms::ConnectionFactory::~~ConnectionFactory () [inline, virtual]`

6.231.3 Member Function Documentation

6.231.3.1 `static ConnectionFactory* cms::ConnectionFactory::createCMSConnectionFactory (const std::string & brokerURI) throw (cms::CMSException) [static]`

Static method that is used to create a provider specific connection factory.

The provider implements this method in their library and returns an instance of a **ConnectionFactory** (p. 1228) derived object. Clients can use this method to remain abstracted from the specific CMS implementation being used.

Parameters

brokerURI The remote address to use to connect to the Provider.

Returns

A pointer to a provider specific implementation of the **ConnectionFactory** (p. 1228) interface, the caller is responsible for deleting this resource.

Exceptions

CMSException (p. 1074) if an internal error occurs while creating the **ConnectionFactory** (p. 1228).

6.231.3.2 `virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & username, const std::string & password) throw (cms::CMSException) [pure virtual]`

Creates a connection with the default specified identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3356) method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless `createConnection` will continue to use the default values that were set in the Constructor.

Parameters

username The user name used to authenticate with the Provider.

password The password used to authenticate with the Provider.

Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions

CMSEException (p. 1074) if an internal error occurs while creating the **Connection** (p. 1168).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 255).

6.231.3.3 `virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & username, const std::string & password, const std::string & clientId) throw (cms::CMSEException) [pure virtual]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3356) method is explicitly called. The user name and password values passed here do not change the defaults, subsequent calls to the parameterless `createConnection` will continue to use the default values that were set in the Constructor.

Parameters

username The user name used to authenticate with the Provider.

password The password used to authenticate with the Provider.

clientId The Client Id assigned to connection. If the id is the empty string ("") then a random client Id is created for this connection.

Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions

CMSEException (p. 1074) if an internal error occurs while creating the **Connection** (p. 1168).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 255).

6.231.3.4 `virtual Connection* cms::ConnectionFactory::createConnection () throw (CMSEException) [pure virtual]`

Creates a connection with the default user identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3356) method is explicitly called.

Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions

CMSException (p. 1074) if an internal error occurs while creating the **Connection** (p. 1168).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 254).

The documentation for this class was generated from the following file:

- `src/main/cms/ConnectionFactory.h`

6.232 activemq::commands::ConnectionId Class Reference

```
#include <src/main/activemq/commands/ConnectionId.h>
```

Inheritance diagram for `activemq::commands::ConnectionId`:

Public Types

- `typedef decaf::lang::PointerComparator< ConnectionId > COMPARATOR`

Public Member Functions

- **ConnectionId** ()
- **ConnectionId** (const **ConnectionId** &other)
- **ConnectionId** (const **SessionId** *sessionId)
- **ConnectionId** (const **ProducerId** *producerId)
- **ConnectionId** (const **ConsumerId** *consumerId)
- virtual **~ConnectionId** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaller share.
- virtual **ConnectionId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)

- virtual int **compareTo** (const **ConnectionId** &value) const
- virtual bool **equals** (const **ConnectionId** &value) const
- virtual bool **operator==** (const **ConnectionId** &value) const
- virtual bool **operator<** (const **ConnectionId** &value) const
- **ConnectionId** & **operator=** (const **ConnectionId** &other)

Static Public Attributes

- static const unsigned char **ID_CONNECTIONID** = 120

Protected Attributes

- std::string **value**

6.232.1 Member Typedef Documentation

- 6.232.1.1** typedef decaf::lang::PointerComparator<ConnectionId>
activemq::commands::ConnectionId::COMPARATOR

6.232.2 Constructor & Destructor Documentation

- 6.232.2.1** activemq::commands::ConnectionId::ConnectionId ()
- 6.232.2.2** activemq::commands::ConnectionId::ConnectionId (const **ConnectionId**
& *other*)
- 6.232.2.3** activemq::commands::ConnectionId::ConnectionId (const **SessionId** *
sessionId)
- 6.232.2.4** activemq::commands::ConnectionId::ConnectionId (const **ProducerId** *
producerId)
- 6.232.2.5** activemq::commands::ConnectionId::ConnectionId (const **ConsumerId** *
consumerId)
- 6.232.2.6** virtual activemq::commands::ConnectionId::~~ConnectionId ()
[virtual]

6.232.3 Member Function Documentation

- 6.232.3.1** virtual **ConnectionId*** ac-
tivemq::commands::ConnectionId::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.232.3.2 `virtual int activemq::commands::ConnectionId::compareTo (const
ConnectionId & value) const` [virtual]

6.232.3.3 `virtual void activemq::commands::ConnectionId::copyDataStructure (const
DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.232.3.4 `virtual bool activemq::commands::ConnectionId::equals (const
DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.232.3.5 `virtual bool activemq::commands::ConnectionId::equals (const
ConnectionId & value) const` [virtual]

6.232.3.6 `virtual unsigned char ac-
tivemq::commands::ConnectionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

- 6.232.3.7 `virtual const std::string& activemq::commands::ConnectionId::getValue () const [virtual]`
- 6.232.3.8 `virtual std::string& activemq::commands::ConnectionId::getValue () [virtual]`
- 6.232.3.9 `virtual bool activemq::commands::ConnectionId::operator< (const ConnectionId & value) const [virtual]`
- 6.232.3.10 `ConnectionId& activemq::commands::ConnectionId::operator= (const ConnectionId & other)`
- 6.232.3.11 `virtual bool activemq::commands::ConnectionId::operator== (const ConnectionId & value) const [virtual]`
- 6.232.3.12 `virtual void activemq::commands::ConnectionId::setValue (const std::string & value) [virtual]`
- 6.232.3.13 `virtual std::string activemq::commands::ConnectionId::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.232.4 Field Documentation

- 6.232.4.1 `const unsigned char activemq::commands::ConnectionId::ID_ - CONNECTIONID = 120 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

- 6.232.4.2 `std::string activemq::commands::ConnectionId::value [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionId.h`

6.233 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1234).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller`:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.233.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1234). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.233.2 Constructor & Destructor Documentation

6.233.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

6.233.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

6.233.3 Member Function Documentation

6.233.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.233.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.233.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.233.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.233.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.233.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.233.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h`

6.234 `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionIdMarshaller` (p. 1238).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller`:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.234.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1238). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.234.2 Constructor & Destructor Documentation

6.234.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::ConnectionIdMarshaller () [inline]`

6.234.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::~~ConnectionIdMarshaller () [inline, virtual]`

6.234.3 Member Function Documentation

6.234.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.234.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.234.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.234.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.234.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.234.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.234.3.7 virtual void `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightUnmarshal`
(`OpenWireFormat` * *wireFormat*, `commands::DataStructure`
* *dataStructure*, `decaf::io::DataInputStream` * *dataIn*,
`utils::BooleanStream` * *bs*) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h`

6.235 `activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionIdMarshaller` (p. 1241).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller`:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.235.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1241). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.235.2 Constructor & Destructor Documentation

6.235.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::ConnectionIdMarshaller () [inline]`

6.235.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::~~ConnectionIdMarshaller () [inline, virtual]`

6.235.3 Member Function Documentation

6.235.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.235.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.235.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.235.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.235.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.235.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.235.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h`

6.236 `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionIdMarshaller` (p. 1245).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller`:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.236.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1245). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.236.2 Constructor & Destructor Documentation

6.236.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::ConnectionIdMarshaller () [inline]`

6.236.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::~~ConnectionIdMarshaller () [inline, virtual]`

6.236.3 Member Function Documentation

6.236.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.236.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.236.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.236.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.236.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.236.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.236.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h

6.237 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1249).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.237.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1249). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.237.2 Constructor & Destructor Documentation

6.237.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::ConnectionIdMarshaller () [inline]`

6.237.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::~~ConnectionIdMarshaller () [inline, virtual]`

6.237.3 Member Function Documentation

6.237.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.237.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.237.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.237.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.237.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.237.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.237.3.7 virtual void **activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h`

6.238 **activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1253).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller**:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.238.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1253). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.238.2 Constructor & Destructor Documentation

6.238.2.1 `activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::ConnectionIdMarshaller () [inline]`

6.238.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::~~ConnectionIdMarshaller () [inline, virtual]`

6.238.3 Member Function Documentation

6.238.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.238.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.238.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.238.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.238.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.238.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.238.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h

6.239 activemq::commands::ConnectionInfo Class Reference

```
#include <src/main/activemq/commands/ConnectionInfo.h>
```

Inheritance diagram for **activemq::commands::ConnectionInfo**:

Public Member Functions

- **ConnectionInfo** ()
- virtual **~ConnectionInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **ConnectionInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer< ConnectionId > & getConnectionId** () const
- virtual **Pointer< ConnectionId > & getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer< ConnectionId > &connectionId**)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const std::string & **getPassword** () const
- virtual std::string & **getPassword** ()
- virtual void **setPassword** (const std::string &password)
- virtual const std::string & **getUserName** () const
- virtual std::string & **getUserName** ()
- virtual void **setUserName** (const std::string &userName)
- virtual const std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer< BrokerId > > &brokerPath**)
- virtual bool **isBrokerMasterConnector** () const
- virtual void **setBrokerMasterConnector** (bool brokerMasterConnector)
- virtual bool **isManageable** () const
- virtual void **setManageable** (bool manageable)
- virtual bool **isClientMaster** () const
- virtual void **setClientMaster** (bool clientMaster)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool faultTolerant)
- virtual bool **isConnectionInfo** () const
- virtual **Pointer< Command > visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONINFO** = 3

Protected Attributes

- **Pointer< ConnectionId > connectionId**
- std::string **clientId**
- std::string **password**
- std::string **userName**
- std::vector< decaf::lang::Pointer< BrokerId > > **brokerPath**
- bool **brokerMasterConnector**
- bool **manageable**
- bool **clientMaster**
- bool **faultTolerant**

6.239.1 Constructor & Destructor Documentation

6.239.1.1 `activemq::commands::ConnectionInfo::ConnectionInfo ()`

6.239.1.2 `virtual activemq::commands::ConnectionInfo::~~ConnectionInfo ()`
[virtual]

6.239.2 Member Function Documentation

6.239.2.1 `virtual ConnectionInfo* activemq::commands::ConnectionInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.239.2.2 `virtual void activemq::commands::ConnectionInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.239.2.3 `Pointer<RemoveInfo> activemq::commands::ConnectionInfo::createRemoveCommand () const`

6.239.2.4 `virtual bool activemq::commands::ConnectionInfo::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.239.2.5 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath () const [virtual]`

6.239.2.6 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath () [virtual]`

6.239.2.7 `virtual const std::string& activemq::commands::ConnectionInfo::getClientId () const [virtual]`

6.239.2.8 `virtual std::string& activemq::commands::ConnectionInfo::getClientId () [virtual]`

6.239.2.9 `virtual Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId () [virtual]`

6.239.2.10 `virtual const Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId () const [virtual]`

6.239.2.11 `virtual unsigned char activemq::commands::ConnectionInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.239.2.12** `virtual const std::string& activemq::commands::ConnectionInfo::getPassword () const [virtual]`
- 6.239.2.13** `virtual std::string& activemq::commands::ConnectionInfo::getPassword () [virtual]`
- 6.239.2.14** `virtual const std::string& activemq::commands::ConnectionInfo::getUserName () const [virtual]`
- 6.239.2.15** `virtual std::string& activemq::commands::ConnectionInfo::getUserName () [virtual]`
- 6.239.2.16** `virtual bool activemq::commands::ConnectionInfo::isBrokerMasterConnector () const [virtual]`
- 6.239.2.17** `virtual bool activemq::commands::ConnectionInfo::isClientMaster () const [virtual]`
- 6.239.2.18** `virtual bool activemq::commands::ConnectionInfo::isConnectionInfo () const [inline, virtual]`

Returns

an answer of true to the `isConnectionInfo()` (p. 1261) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 698).

- 6.239.2.19 `virtual bool activemq::commands::ConnectionInfo::isFaultTolerant ()`
`const [virtual]`
- 6.239.2.20 `virtual bool activemq::commands::ConnectionInfo::isManageable ()`
`const [virtual]`
- 6.239.2.21 `virtual void ac-`
`tivemq::commands::ConnectionInfo::setBrokerMasterConnector (bool`
`brokerMasterConnector) [virtual]`
- 6.239.2.22 `virtual void activemq::commands::ConnectionInfo::setBrokerPath (`
`const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)`
`[virtual]`
- 6.239.2.23 `virtual void activemq::commands::ConnectionInfo::setClientId (const`
`std::string & clientId) [virtual]`
- 6.239.2.24 `virtual void activemq::commands::ConnectionInfo::setClientMaster (`
`bool clientMaster) [virtual]`
- 6.239.2.25 `virtual void activemq::commands::ConnectionInfo::setConnectionId (`
`const Pointer< ConnectionId > & connectionId) [virtual]`
- 6.239.2.26 `virtual void activemq::commands::ConnectionInfo::setFaultTolerant (`
`bool faultTolerant) [virtual]`
- 6.239.2.27 `virtual void activemq::commands::ConnectionInfo::setManageable (`
`bool manageable) [virtual]`
- 6.239.2.28 `virtual void activemq::commands::ConnectionInfo::setPassword (const`
`std::string & password) [virtual]`
- 6.239.2.29 `virtual void activemq::commands::ConnectionInfo::setUserName (`
`const std::string & userName) [virtual]`
- 6.239.2.30 `virtual std::string activemq::commands::ConnectionInfo::toString ()`
`const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

- 6.239.2.31 `virtual Pointer<Command> activemq::commands::ConnectionInfo::visit`
`(activemq::state::CommandVisitor * visitor) throw (`
`exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.239.3 Field Documentation

- 6.239.3.1** `bool activemq::commands::ConnectionInfo::brokerMasterConnector` [protected]
- 6.239.3.2** `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::ConnectionInfo::brokerPath` [protected]
- 6.239.3.3** `std::string activemq::commands::ConnectionInfo::clientId` [protected]
- 6.239.3.4** `bool activemq::commands::ConnectionInfo::clientMaster` [protected]
- 6.239.3.5** `Pointer<ConnectionId>` `activemq::commands::ConnectionInfo::connectionId` [protected]
- 6.239.3.6** `bool activemq::commands::ConnectionInfo::faultTolerant` [protected]
- 6.239.3.7** `const unsigned char activemq::commands::ConnectionInfo::ID _ - CONNECTIONINFO = 3` [static]
- 6.239.3.8** `bool activemq::commands::ConnectionInfo::manageable` [protected]
- 6.239.3.9** `std::string activemq::commands::ConnectionInfo::password` [protected]
- 6.239.3.10** `std::string activemq::commands::ConnectionInfo::userName` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionInfo.h`

6.240 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1263).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller`:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.240.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1263). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.240.2 Constructor & Destructor Documentation

6.240.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::ConnectionInfoMarshaller () [inline]`

6.240.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller () [inline, virtual]`

6.240.3 Member Function Documentation

6.240.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.240.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.240.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.240.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

6.240.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

6.240.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 739).

```
6.240.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h

6.241 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1267).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller**:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.241.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1267). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.241.2 Constructor & Destructor Documentation

6.241.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::ConnectionInfoMarshaller()` `[inline]`

6.241.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller()` `[inline, virtual]`

6.241.3 Member Function Documentation

6.241.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.241.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.241.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.241.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.241.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

6.241.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 706).

```
6.241.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h`

6.242 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1271).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller**:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.242.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1271). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.242.2 Constructor & Destructor Documentation

6.242.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::ConnectionInfoM
() [inline]`

6.242.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::~~ConnectionInfo
() [inline, virtual]`

6.242.3 Member Function Documentation

6.242.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.242.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::getDataStructureT
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.242.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.242.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

6.242.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

6.242.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 712).

6.242.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h

6.243 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1275).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.243.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1275). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.243.2 Constructor & Destructor Documentation

6.243.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::ConnectionInfoMarshaller()` `[inline]`

6.243.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller()` `[inline, virtual]`

6.243.3 Member Function Documentation

6.243.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.243.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.243.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.243.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.243.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

6.243.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

6.243.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h`

6.244 `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionInfoMarshaller` (p. 1279).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller`:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.244.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1279). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.244.2 Constructor & Destructor Documentation

6.244.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::ConnectionInfoMarshaller () [inline]`

6.244.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller () [inline, virtual]`

6.244.3 Member Function Documentation

6.244.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.244.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.244.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.244.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

6.244.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

6.244.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.244.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h`

6.245 `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionInfoMarshaller` (p. 1283).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller`:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.245.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1283). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.245.2 Constructor & Destructor Documentation

6.245.2.1 `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::ConnectionInfoMarshaller () [inline]`

6.245.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller () [inline, virtual]`

6.245.3 Member Function Documentation

6.245.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.245.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.245.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.245.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

6.245.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

6.245.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 732).

6.245.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionInfoMarshaller.h**

6.246 cms::ConnectionMetaData Class Reference

A **ConnectionMetaData** (p. 1287) object provides information describing the **Connection** (p. 1168) object.

```
#include <src/main/cms/ConnectionMetaData.h>
```

Inheritance diagram for cms::ConnectionMetaData:

Public Member Functions

- virtual `~ConnectionMetaData ()`
- virtual `std::string getCMSVersion () const =0 throw (cms::CMSEException)`
Gets the CMS API version.
- virtual `int getCMSMajorVersion () const =0 throw (cms::CMSEException)`
Gets the CMS major version number.
- virtual `int getCMSMinorVersion () const =0 throw (cms::CMSEException)`
Gets the CMS minor version number.
- virtual `std::string getCMSProviderName () const =0 throw (cms::CMSEException)`
Gets the CMS provider name.
- virtual `std::string getProviderVersion () const =0 throw (cms::CMSEException)`
Gets the CMS provider version.
- virtual `int getProviderMajorVersion () const =0 throw (cms::CMSEException)`
Gets the CMS provider major version number.
- virtual `int getProviderMinorVersion () const =0 throw (cms::CMSEException)`
Gets the CMS provider minor version number.
- virtual `std::vector< std::string > getCMSXPropertyNames () const =0 throw (cms::CMSEException)`
Gets an Vector of the CMSX property names.

6.246.1 Detailed Description

A **ConnectionMetaData** (p.1287) object provides information describing the **Connection** (p.1168) object.

Since

1.3

6.246.2 Constructor & Destructor Documentation

- 6.246.2.1** `virtual cms::ConnectionMetaData::~~ConnectionMetaData () [inline, virtual]`

6.246.3 Member Function Documentation

- 6.246.3.1** `virtual int cms::ConnectionMetaData::getCMSMajorVersion () const throw (cms::CMSEException) [pure virtual]`

Gets the CMS major version number.

Returns

the CMS API major version number

Exceptions

CMSEException (p. 1074) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 263).

6.246.3.2 `virtual int cms::ConnectionMetaData::getCMSMinorVersion () const throw (cms::CMSEException) [pure virtual]`

Gets the CMS minor version number.

Returns

the CMS API minor version number

Exceptions

CMSEException (p. 1074) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 263).

6.246.3.3 `virtual std::string cms::ConnectionMetaData::getCMSProviderName () const throw (cms::CMSEException) [pure virtual]`

Gets the CMS provider name.

Returns

the CMS provider name

Exceptions

CMSEException (p. 1074) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 263).

6.246.3.4 `virtual std::string cms::ConnectionMetaData::getCMSVersion () const throw (cms::CMSEException) [pure virtual]`

Gets the CMS API version.

Returns

the CMS API Version in String form.

Exceptions

CMSEException (p. 1074) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 264).

6.246.3.5 `virtual std::vector<std::string>
cms::ConnectionMetaData::getCMSXPropertyNames (
) const throw (cms::CMSEException) [pure virtual]`

Gets an Vector of the CMSX property names.

Returns

an Vector of CMSX property names

Exceptions

CMSEException (p. 1074) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 264).

6.246.3.6 `virtual int cms::ConnectionMetaData::getProviderMajorVersion ()
const throw (cms::CMSEException) [pure virtual]`

Gets the CMS provider major version number.

Returns

the CMS provider major version number

Exceptions

CMSEException (p. 1074) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 264).

6.246.3.7 `virtual int cms::ConnectionMetaData::getProviderMinorVersion ()
const throw (cms::CMSEException) [pure virtual]`

Gets the CMS provider minor version number.

Returns

the CMS provider minor version number

Exceptions

CMSEException (p. 1074) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 265).

6.246.3.8 `virtual std::string cms::ConnectionMetaData::getProviderVersion ()
const throw (cms::CMSEException) [pure virtual]`

Gets the CMS provider version.

Returns

the CMS provider version

Exceptions

CMSException (p. 1074) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 265).

The documentation for this class was generated from the following file:

- `src/main/cms/ConnectionMetaData.h`

6.247 activemq::state::ConnectionState Class Reference

```
#include <src/main/activemq/state/ConnectionState.h>
```

Public Member Functions

- **ConnectionState** (const **Pointer**< **ConnectionInfo** > &info)
- virtual **~ConnectionState** ()
- **std::string toString** () const
- const **Pointer**< **commands::ConnectionInfo** > & **getInfo** () const
- void **checkShutdown** () const
- void **shutdown** ()
- void **reset** (const **Pointer**< **ConnectionInfo** > &info)
- void **addTempDestination** (const **Pointer**< **DestinationInfo** > &info)
- void **removeTempDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- void **addTransactionState** (const **Pointer**< **TransactionId** > &id)
- const **Pointer**< **TransactionState** > & **getTransactionState** (const **Pointer**< **TransactionId** > &id) const
- **std::vector**< **Pointer**< **TransactionState** > > **getTransactionStates** () const
- **Pointer**< **TransactionState** > **removeTransactionState** (const **Pointer**< **TransactionId** > &id)
- void **addSession** (const **Pointer**< **SessionInfo** > &info)
- **Pointer**< **SessionState** > **removeSession** (const **Pointer**< **SessionId** > &id)
- const **Pointer**< **SessionState** > & **getSessionState** (const **Pointer**< **SessionId** > &id) const
- const **StlList**< **Pointer**< **DestinationInfo** > > & **getTempDesinations** () const
- **std::vector**< **Pointer**< **SessionState** > > **getSessionStates** () const
- **StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > **getRecoveringPullConsumers** ()
- void **setConnectionInterruptProcessingComplete** (bool connectionInterruptProcessingComplete)
- bool **isConnectionInterruptProcessingComplete** ()

6.247.1 Constructor & Destructor Documentation

6.247.1.1 `activemq::state::ConnectionState::ConnectionState (const Pointer< ConnectionInfo > & info)`

6.247.1.2 `virtual activemq::state::ConnectionState::~~ConnectionState ()`
[virtual]

6.247.2 Member Function Documentation

6.247.2.1 `void activemq::state::ConnectionState::addSession (const Pointer< SessionInfo > & info)` [inline]

6.247.2.2 `void activemq::state::ConnectionState::addTempDestination (const Pointer< DestinationInfo > & info)` [inline]

6.247.2.3 `void activemq::state::ConnectionState::addTransactionState (const Pointer< TransactionId > & id)` [inline]

6.247.2.4 `void activemq::state::ConnectionState::checkShutdown ()` const

6.247.2.5 `const Pointer<commands::ConnectionInfo>& activemq::state::ConnectionState::getInfo ()` const [inline]

6.247.2.6 `StlMap< Pointer<ConsumerId>, Pointer<ConsumerInfo>, ConsumerId::COMPARATOR > & activemq::state::ConnectionState::getRecoveringPullConsumers ()` [inline]

6.247.2.7 `const Pointer<SessionState>& activemq::state::ConnectionState::getSessionState (const Pointer< SessionId > & id)` const [inline]

6.247.2.8 `std::vector< Pointer<SessionState> > & activemq::state::ConnectionState::getSessionStates ()` const [inline]

6.247.2.9 `const StlList< Pointer<DestinationInfo> >& activemq::state::ConnectionState::getTempDesinations ()` const [inline]

6.247.2.10 `const Pointer<TransactionState>& activemq::state::ConnectionState::getTransactionState (const Pointer< TransactionId > & id)` const [inline]

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

- 6.247.2.11 `std::vector< Pointer<TransactionState> > activemq::state::ConnectionState::getTransactionStates () const` [inline]
- 6.247.2.12 `bool activemq::state::ConnectionState::isConnectionInterruptProcessingComplete ()` [inline]
- 6.247.2.13 `Pointer<SessionState> activemq::state::ConnectionState::removeSession (const Pointer< SessionId > & id)` [inline]
- 6.247.2.14 `void activemq::state::ConnectionState::removeTempDestination (const Pointer< ActiveMQDestination > & destination)` [inline]

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

- 6.247.2.15 `Pointer<TransactionState> activemq::state::ConnectionState::removeTransactionState (const Pointer< TransactionId > & id)` [inline]
- 6.247.2.16 `void activemq::state::ConnectionState::reset (const Pointer< ConnectionInfo > & info)`
- 6.247.2.17 `void activemq::state::ConnectionState::setConnectionInterruptProcessingComplete (bool connectionInterruptProcessingComplete)` [inline]
- 6.247.2.18 `void activemq::state::ConnectionState::shutdown ()`
- 6.247.2.19 `std::string activemq::state::ConnectionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConnectionState.h`

6.248 activemq::state::ConnectionStateTracker Class Reference

```
#include <src/main/activemq/state/ConnectionStateTracker.h>
```

Inheritance diagram for `activemq::state::ConnectionStateTracker`:

Public Member Functions

- `ConnectionStateTracker ()`
- `virtual ~ConnectionStateTracker ()`
- `Pointer< Tracked > track (const Pointer< Command > &command) throw (decaf::io::IOException)`

- void **trackBack** (const **Pointer**< **Command** > &command)
- void **restore** (const **Pointer**< **transport::Transport** > &transport) throw (decaf::io::IOException)
- void **connectionInterruptProcessingComplete** (**transport::Transport** *transport, const **Pointer**< **ConnectionId** > &connectionId)
- void **transportInterrupted** ()
- virtual **Pointer**< **Command** > **processDestinationInfo** (**DestinationInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processRemoveDestination** (**DestinationInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processProducerInfo** (**ProducerInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processRemoveProducer** (**ProducerId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processConsumerInfo** (**ConsumerInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processRemoveConsumer** (**ConsumerId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processSessionInfo** (**SessionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processRemoveSession** (**SessionId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processConnectionInfo** (**ConnectionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processRemoveConnection** (**ConnectionId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processMessage** (**Message** *message) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processMessageAck** (**MessageAck** *ack) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processBeginTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processPrepareTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processCommitTransactionOnePhase** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processCommitTransactionTwoPhase** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processRollbackTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer**< **Command** > **processEndTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- bool **isRestoreConsumers** () const
- void **setRestoreConsumers** (bool restoreConsumers)
- bool **isRestoreProducers** () const
- void **setRestoreProducers** (bool restoreProducers)
- bool **isRestoreSessions** () const
- void **setRestoreSessions** (bool restoreSessions)
- bool **isTrackTransactions** () const
- void **setTrackTransactions** (bool trackTransactions)
- bool **isRestoreTransaction** () const

- void **setRestoreTransaction** (bool restoreTransaction)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool trackMessages)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int maxCacheSize)
- bool **isTrackTransactionProducers** () const
- void **setTrackTransactionProducers** (bool trackTransactionProducers)

Friends

- class **RemoveTransactionAction**

6.248.1 Constructor & Destructor Documentation

6.248.1.1 `activemq::state::ConnectionStateTracker::ConnectionStateTracker ()`

6.248.1.2 `virtual
activemq::state::ConnectionStateTracker::~~ConnectionStateTracker ()
[virtual]`

6.248.2 Member Function Documentation

6.248.2.1 `void ac-
tivemq::state::ConnectionStateTracker::connectionInterruptProcessingComplete
(transport::Transport * transport, const Pointer< ConnectionId > &
connectionId)`

6.248.2.2 `int activemq::state::ConnectionStateTracker::getMaxCacheSize ()
const [inline]`

6.248.2.3 `bool activemq::state::ConnectionStateTracker::isRestoreConsumers ()
const [inline]`

6.248.2.4 `bool activemq::state::ConnectionStateTracker::isRestoreProducers ()
const [inline]`

6.248.2.5 `bool activemq::state::ConnectionStateTracker::isRestoreSessions ()
const [inline]`

6.248.2.6 `bool activemq::state::ConnectionStateTracker::isRestoreTransaction ()
const [inline]`

6.248.2.7 `bool activemq::state::ConnectionStateTracker::isTrackMessages ()
const [inline]`

6.248.2.8 `bool ac-
tivemq::state::ConnectionStateTracker::isTrackTransactionProducers ()
const [inline]`

6.248.2.9 `bool activemq::state::ConnectionStateTracker::isTrackTransactions ()
const [inline]`

6.248.2.10 `virtual Pointer<Command> ac-
tivemq::state::ConnectionStateTracker::processBeginTransaction ()
TransactionInfo * info) throw (exceptions::ActiveMQException)
[virtual]`

Implements `activemq::state::CommandVisitor` (p. 1115).

6.248.2.11 `virtual Pointer<Command> ac-
tivemq::state::ConnectionStateTracker::processCommitTransactionOnePhase
(TransactionInfo * info) throw (exceptions::ActiveMQException)
[virtual]`

Implements `activemq::state::CommandVisitor` (p. 1115).

6.248.2.12 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionTwoPhase (TransactionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1115).

6.248.2.13 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConnectionInfo (ConnectionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1116).

6.248.2.14 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConsumerInfo (ConsumerInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1116).

6.248.2.15 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processDestinationInfo (DestinationInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1116).

6.248.2.16 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processEndTransaction (TransactionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1116).

6.248.2.17 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessage (Message * *message*) throw (exceptions::ActiveMQException)
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1117).

6.248.2.18 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessageAck (MessageAck * *ack*) throw (exceptions::ActiveMQException)
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1117).

6.248.2.19 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processPrepareTransaction (TransactionInfo * info) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1117).

6.248.2.20 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processProducerInfo (ProducerInfo * info) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1118).

6.248.2.21 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConnection (ConnectionId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1118).

6.248.2.22 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConsumer (ConsumerId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1118).

6.248.2.23 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveDestination (DestinationInfo * info) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1118).

6.248.2.24 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveProducer (ProducerId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1118).

6.248.2.25 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveSession (SessionId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1119).

6.248.2.26 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRollbackTransaction (TransactionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1119).

6.248.2.27 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processSessionInfo (SessionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1119).

- 6.248.2.28 `void activemq::state::ConnectionStateTracker::restore (const Pointer< transport::Transport > & transport) throw (decaf::io::IOException)`
- 6.248.2.29 `void activemq::state::ConnectionStateTracker::setMaxCacheSize (int maxCacheSize) [inline]`
- 6.248.2.30 `void activemq::state::ConnectionStateTracker::setRestoreConsumers (bool restoreConsumers) [inline]`
- 6.248.2.31 `void activemq::state::ConnectionStateTracker::setRestoreProducers (bool restoreProducers) [inline]`
- 6.248.2.32 `void activemq::state::ConnectionStateTracker::setRestoreSessions (bool restoreSessions) [inline]`
- 6.248.2.33 `void activemq::state::ConnectionStateTracker::setRestoreTransaction (bool restoreTransaction) [inline]`
- 6.248.2.34 `void activemq::state::ConnectionStateTracker::setTrackMessages (bool trackMessages) [inline]`
- 6.248.2.35 `void activemq::state::ConnectionStateTracker::setTrackTransactionProducers (bool trackTransactionProducers) [inline]`
- 6.248.2.36 `void activemq::state::ConnectionStateTracker::setTrackTransactions (bool trackTransactions) [inline]`
- 6.248.2.37 `Pointer<Tracked> activemq::state::ConnectionStateTracker::track (const Pointer< Command > & command) throw (decaf::io::IOException)`
- 6.248.2.38 `void activemq::state::ConnectionStateTracker::trackBack (const Pointer< Command > & command)`
- 6.248.2.39 `void activemq::state::ConnectionStateTracker::transportInterrupted ()`

6.248.3 Friends And Related Function Documentation

- 6.248.3.1 `friend class RemoveTransactionAction` [friend]

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConnectionStateTracker.h`

6.249 decaf::util::logging::ConsoleHandler Class Reference

This **Handler** (p. 1852) publishes log records to System.err.

```
#include <src/main/decaf/util/logging/ConsoleHandler.h>
```

Inheritance diagram for decaf::util::logging::ConsoleHandler:

Public Member Functions

- **ConsoleHandler** ()
- virtual **~ConsoleHandler** ()
- virtual void **close** () throw (decaf::io::IOException)
Close the current output stream.
- virtual void **publish** (const **LogRecord** &record)
*Publish the Log Record to this **Handler** (p. 1852).*

6.249.1 Detailed Description

This **Handler** (p. 1852) publishes log records to System.err. By default the **SimpleFormatter** (p. 3278) is used to generate brief summaries.

Configuration: By default each **ConsoleHandler** (p. 1300) is initialized using the following **Log-Manager** (p. 2254) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

ConsoleHandler.level specifies the default level for the **Handler** (p. 1852) (defaults to **Level.INFO** (p. 2190)). ConsoleHandler.filter specifies the name of a **Filter** (p. 1770) class to use (defaults to no **Filter** (p. 1770)). ConsoleHandler.formatter specifies the name of a **Formatter** (p. 1838) class to use (defaults to **SimpleFormatter** (p. 3278)).

Since

1.0

6.249.2 Constructor & Destructor Documentation

6.249.2.1 decaf::util::logging::ConsoleHandler::ConsoleHandler ()

6.249.2.2 virtual decaf::util::logging::ConsoleHandler::~~ConsoleHandler ()
 [inline, virtual]

6.249.3 Member Function Documentation

6.249.3.1 virtual void decaf::util::logging::ConsoleHandler::close () throw (decaf::io::IOException) [virtual]

Close the current output stream.

Override the **StreamHandler** (p. 3412) close to flush the Std Err stream but doesn't close.

Exceptions

IOException

Reimplemented from decaf::util::logging::StreamHandler (p. 3414).

6.249.3.2 virtual void decaf::util::logging::ConsoleHandler::publish (const LogRecord & *record*) [virtual]

Publish the Log Record to this **Handler** (p. 1852).

Parameters

record The LogRecord (p. 2261) to Publish

Reimplemented from **decaf::util::logging::StreamHandler** (p. 3415).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**ConsoleHandler.h**

6.250 activemq::commands::ConsumerControl Class Reference

```
#include <src/main/activemq/commands/ConsumerControl.h>
```

Inheritance diagram for activemq::commands::ConsumerControl:

Public Member Functions

- **ConsumerControl** ()
- virtual **~ConsumerControl** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConsumerControl * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > &**getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > &**getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual const **Pointer**< **ConsumerId** > &**getConsumerId** () const

- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual int **getPrefetch** () const
- virtual void **setPrefetch** (int prefetch)
- virtual bool **isFlush** () const
- virtual void **setFlush** (bool flush)
- virtual bool **isStart** () const
- virtual void **setStart** (bool start)
- virtual bool **isStop** () const
- virtual void **setStop** (bool stop)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _CONSUMERCONTROL** = 17

Protected Attributes

- **Pointer**< **ActiveMQDestination** > destination
- bool close
- **Pointer**< **ConsumerId** > consumerId
- int prefetch
- bool flush
- bool start
- bool stop

6.250.1 Constructor & Destructor Documentation

6.250.1.1 **activemq::commands::ConsumerControl::ConsumerControl** ()

6.250.1.2 **virtual activemq::commands::ConsumerControl::~~ConsumerControl** ()
[virtual]

6.250.2 Member Function Documentation

6.250.2.1 **virtual ConsumerControl* activemq::commands::ConsumerControl::cloneDataStructure**
() const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.250.2.2 `virtual void activemq::commands::ConsumerControl::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.250.2.3 `virtual bool activemq::commands::ConsumerControl::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.250.2.4 `virtual const Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId () const [virtual]`

6.250.2.5 `virtual Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId () [virtual]`

6.250.2.6 `virtual unsigned char activemq::commands::ConsumerControl::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements `activemq::commands::DataStructure` (p. 1557).

- 6.250.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerControl::getDestination () const`
[virtual]
- 6.250.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerControl::getDestination ()`
[virtual]
- 6.250.2.9 `virtual int activemq::commands::ConsumerControl::getPrefetch () const`
[virtual]
- 6.250.2.10 `virtual bool activemq::commands::ConsumerControl::isClose () const`
[virtual]
- 6.250.2.11 `virtual bool activemq::commands::ConsumerControl::isFlush () const`
[virtual]
- 6.250.2.12 `virtual bool activemq::commands::ConsumerControl::isStart () const`
[virtual]
- 6.250.2.13 `virtual bool activemq::commands::ConsumerControl::isStop () const`
[virtual]
- 6.250.2.14 `virtual void activemq::commands::ConsumerControl::setClose (bool close)`
[virtual]
- 6.250.2.15 `virtual void activemq::commands::ConsumerControl::setConsumerId (const Pointer< ConsumerId > & consumerId)`
[virtual]
- 6.250.2.16 `virtual void activemq::commands::ConsumerControl::setDestination (const Pointer< ActiveMQDestination > & destination)`
[virtual]
- 6.250.2.17 `virtual void activemq::commands::ConsumerControl::setFlush (bool flush)`
[virtual]
- 6.250.2.18 `virtual void activemq::commands::ConsumerControl::setPrefetch (int prefetch)`
[virtual]
- 6.250.2.19 `virtual void activemq::commands::ConsumerControl::setStart (bool start)`
[virtual]
- 6.250.2.20 `virtual void activemq::commands::ConsumerControl::setStop (bool stop)`
[virtual]
- 6.250.2.21 `virtual std::string activemq::commands::ConsumerControl::toString () const`
[virtual]

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 700).

6.250.2.22 `virtual Pointer<Command> activemq::commands::ConsumerControl::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1112).

6.250.3 Field Documentation

6.250.3.1 `bool activemq::commands::ConsumerControl::close [protected]`

6.250.3.2 `Pointer<ConsumerId> activemq::commands::ConsumerControl::consumerId [protected]`

6.250.3.3 `Pointer<ActiveMQDestination> activemq::commands::ConsumerControl::destination [protected]`

6.250.3.4 `bool activemq::commands::ConsumerControl::flush [protected]`

6.250.3.5 `const unsigned char activemq::commands::ConsumerControl::ID_CONSUMERCONTROL = 17 [static]`

6.250.3.6 `int activemq::commands::ConsumerControl::prefetch [protected]`

6.250.3.7 `bool activemq::commands::ConsumerControl::start [protected]`

6.250.3.8 `bool activemq::commands::ConsumerControl::stop [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerControl.h`

6.251 `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerControlMarshaller` (p. 1306).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.251.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1306). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.251.2 Constructor & Destructor Documentation

6.251.2.1 `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::ConsumerControlMarshaller () [inline]`

6.251.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::~~ConsumerControlMarshaller () [inline, virtual]`

6.251.3 Member Function Documentation

6.251.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.251.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.251.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.251.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 737).

6.251.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 738).

6.251.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

```
6.251.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h`

6.252 `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerControlMarshaller` (p. 1310).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller`:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.252.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1310). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.252.2 Constructor & Destructor Documentation

6.252.2.1 `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::ConsumerControlMarshaller () [inline]`

6.252.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::~~ConsumerControlMarshaller () [inline, virtual]`

6.252.3 Member Function Documentation

6.252.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.252.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.252.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.252.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 703).

6.252.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 704).

6.252.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.252.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h`

6.253 `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerControlMarshaller` (p. 1314).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller`:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.253.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1314). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.253.2 Constructor & Destructor Documentation

6.253.2.1 `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::ConsumerControlMarshaller () [inline]`

6.253.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::~~ConsumerControlMarshaller () [inline, virtual]`

6.253.3 Member Function Documentation

6.253.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.253.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.253.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.253.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 710).

6.253.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 711).

6.253.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.253.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h`

6.254 `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerControlMarshaller` (p. 1318).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller`:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.254.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1318). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.254.2 Constructor & Destructor Documentation

6.254.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::ConsumerControlMarshaller () [inline]`

6.254.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::~~ConsumerControlMarshaller () [inline, virtual]`

6.254.3 Member Function Documentation

6.254.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.254.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.254.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.254.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 717).

6.254.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 718).

6.254.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

```
6.254.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h`

6.255 `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerControlMarshaller` (p. 1322).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller`:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.255.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1322). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.255.2 Constructor & Destructor Documentation

6.255.2.1 `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::ConsumerControlMarshaller () [inline]`

6.255.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::~~ConsumerControlMarshaller () [inline, virtual]`

6.255.3 Member Function Documentation

6.255.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.255.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.255.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.255.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 723).

6.255.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 724).

6.255.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.255.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h`

6.256 `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerControlMarshaller` (p. 1326).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller`:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.256.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1326). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.256.2 Constructor & Destructor Documentation

6.256.2.1 `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::ConsumerControlMarshaller () [inline]`

6.256.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::~~ConsumerControlMarshaller () [inline, virtual]`

6.256.3 Member Function Documentation

6.256.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.256.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.256.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.256.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 730).

6.256.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 731).

6.256.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

```
6.256.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h`

6.257 `activemq::commands::ConsumerId` Class Reference

```
#include <src/main/activemq/commands/ConsumerId.h>
```

Inheritance diagram for `activemq::commands::ConsumerId`:

Public Types

- `typedef decaf::lang::PointerComparator< ConsumerId > COMPARATOR`

Public Member Functions

- **ConsumerId** ()
- **ConsumerId** (const **ConsumerId** &other)
- **ConsumerId** (const **SessionId** &sessionId, long long consumerId)
- virtual ~**ConsumerId** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConsumerId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- const **Pointer**< **SessionId** > & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual int **compareTo** (const **ConsumerId** &value) const
- virtual bool **equals** (const **ConsumerId** &value) const
- virtual bool **operator==** (const **ConsumerId** &value) const
- virtual bool **operator<** (const **ConsumerId** &value) const
- **ConsumerId** & **operator=** (const **ConsumerId** &other)

Static Public Attributes

- static const unsigned char **ID_CONSUMERID** = 122

Protected Attributes

- std::string **connectionId**
- long long **sessionId**
- long long **value**

6.257.1 Member Typedef Documentation

6.257.1.1 `typedef decaf::lang::PointerComparator<ConsumerId>
activemq::commands::ConsumerId::COMPARATOR`

6.257.2 Constructor & Destructor Documentation

6.257.2.1 `activemq::commands::ConsumerId::ConsumerId ()`

6.257.2.2 `activemq::commands::ConsumerId::ConsumerId (const ConsumerId &
other)`

6.257.2.3 `activemq::commands::ConsumerId::ConsumerId (const SessionId &
sessionId, long long consumerId)`

6.257.2.4 `virtual activemq::commands::ConsumerId::~~ConsumerId () [virtual]`

6.257.3 Member Function Documentation

6.257.3.1 `virtual ConsumerId* ac-
tivemq::commands::ConsumerId::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.257.3.2 `virtual int activemq::commands::ConsumerId::compareTo (const
ConsumerId & value) const [virtual]`

6.257.3.3 `virtual void activemq::commands::ConsumerId::copyDataStructure ()
const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.257.3.4 `virtual bool activemq::commands::ConsumerId::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

- 6.257.3.5 virtual bool activemq::commands::ConsumerId::equals (const ConsumerId & *value*) const [virtual]
- 6.257.3.6 virtual std::string& activemq::commands::ConsumerId::getConnectionId () [virtual]
- 6.257.3.7 virtual const std::string& activemq::commands::ConsumerId::getConnectionId () const [virtual]
- 6.257.3.8 virtual unsigned char activemq::commands::ConsumerId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

- 6.257.3.9 const Pointer<SessionId>& activemq::commands::ConsumerId::getParentId () const
- 6.257.3.10 virtual long long activemq::commands::ConsumerId::getSessionId () const [virtual]
- 6.257.3.11 virtual long long activemq::commands::ConsumerId::getValue () const [virtual]
- 6.257.3.12 virtual bool activemq::commands::ConsumerId::operator< (const ConsumerId & *value*) const [virtual]
- 6.257.3.13 ConsumerId& activemq::commands::ConsumerId::operator= (const ConsumerId & *other*)
- 6.257.3.14 virtual bool activemq::commands::ConsumerId::operator== (const ConsumerId & *value*) const [virtual]
- 6.257.3.15 virtual void activemq::commands::ConsumerId::setConnectionId (const std::string & *connectionId*) [virtual]
- 6.257.3.16 virtual void activemq::commands::ConsumerId::setSessionId (long long *sessionId*) [virtual]
- 6.257.3.17 virtual void activemq::commands::ConsumerId::setValue (long long *value*) [virtual]
- 6.257.3.18 virtual std::string activemq::commands::ConsumerId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.257.4 Field Documentation

6.257.4.1 `std::string activemq::commands::ConsumerId::connectionId` [protected]

6.257.4.2 `const unsigned char activemq::commands::ConsumerId::ID_ -
CONSUMERID = 122` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.257.4.3 `long long activemq::commands::ConsumerId::sessionId` [protected]

6.257.4.4 `long long activemq::commands::ConsumerId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerId.h`

6.258 `activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1334).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller`:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.258.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1334). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.258.2 Constructor & Destructor Documentation

6.258.2.1 `activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::ConsumerIdMarshaller ()` [inline]

6.258.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::~~ConsumerIdMarshaller ()` [inline, virtual]

6.258.3 Member Function Documentation

6.258.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::createObject () const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.258.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::getDataStructureType () const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.258.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.258.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.258.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.258.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.258.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h`

6.259 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1338).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.259.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1338). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.259.2 Constructor & Destructor Documentation

6.259.2.1 **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::ConsumerIdMarshaller** () [inline]

6.259.2.2 **virtual**
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::~~ConsumerIdMarshaller () [inline, virtual]

6.259.3 Member Function Documentation

6.259.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.259.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.259.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.259.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.259.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.259.3.6 virtual void **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal2**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.259.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h`

6.260 **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1342).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller**:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.260.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1342). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.260.2 Constructor & Destructor Documentation

6.260.2.1 `activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::ConsumerIdMarshaller () [inline]`

6.260.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::~~ConsumerIdMarshaller () [inline, virtual]`

6.260.3 Member Function Documentation

6.260.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.260.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.260.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

```
6.260.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

```
6.260.3.5 virtual int ac-
tivismq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.260.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.260.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h`

6.261 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for `ConsumerIdMarshaller` (p. 1345).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller`:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.261.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1345). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.261.2 Constructor & Destructor Documentation

6.261.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::ConsumerIdMarshaller()` [inline]

6.261.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` [inline, virtual]

6.261.3 Member Function Documentation

6.261.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.261.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.261.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.261.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.261.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.261.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.261.3.7 virtual void **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h`

6.262 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1349).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller**:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.262.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1349). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.262.2 Constructor & Destructor Documentation

6.262.2.1 `activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::ConsumerIdMarshaller () [inline]`

6.262.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::~~ConsumerIdMarshaller () [inline, virtual]`

6.262.3 Member Function Documentation

6.262.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.262.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.262.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.262.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.262.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.262.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.262.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h

6.263 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1353).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.263.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1353). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.263.2 Constructor & Destructor Documentation

6.263.2.1 `activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::ConsumerIdMarshaller () [inline]`

6.263.2.2 `virtual
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::~~ConsumerIdMarshaller () [inline, virtual]`

6.263.3 Member Function Documentation

6.263.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.263.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.263.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.263.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.263.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.263.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.263.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h

6.264 activemq::commands::ConsumerInfo Class Reference

```
#include <src/main/activemq/commands/ConsumerInfo.h>
```

Inheritance diagram for **activemq::commands::ConsumerInfo**:

Public Member Functions

- **ConsumerInfo** ()
- virtual **~ConsumerInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **ConsumerInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer< ConsumerId > & getConsumerId** () const
- virtual **Pointer< ConsumerId > & getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer< ConsumerId > &consumerId**)
- virtual bool **isBrowser** () const
- virtual void **setBrowser** (bool browser)
- virtual const **Pointer< ActiveMQDestination > & getDestination** () const
- virtual **Pointer< ActiveMQDestination > & getDestination** ()
- virtual void **setDestination** (const **Pointer< ActiveMQDestination > &destination**)
- virtual int **getPrefetchSize** () const
- virtual void **setPrefetchSize** (int prefetchSize)
- virtual int **getMaximumPendingMessageLimit** () const
- virtual void **setMaximumPendingMessageLimit** (int maximumPendingMessageLimit)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool dispatchAsync)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &selector)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual bool **isNoLocal** () const
- virtual void **setNoLocal** (bool noLocal)
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool exclusive)
- virtual bool **isRetroactive** () const
- virtual void **setRetroactive** (bool retroactive)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char priority)
- virtual const std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer< BrokerId > > &brokerPath**)

- virtual const **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** () const
- virtual **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** ()
- virtual void **setAdditionalPredicate** (const **Pointer**< **BooleanExpression** > & **additionalPredicate**)
- virtual bool **isNetworkSubscription** () const
- virtual void **setNetworkSubscription** (bool **networkSubscription**)
- virtual bool **isOptimizedAcknowledge** () const
- virtual void **setOptimizedAcknowledge** (bool **optimizedAcknowledge**)
- virtual bool **isNoRangeAcks** () const
- virtual void **setNoRangeAcks** (bool **noRangeAcks**)
- virtual const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** ()
- virtual void **setNetworkConsumerPath** (const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **networkConsumerPath**)
- virtual bool **isConsumerInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (**exceptions::ActiveMQException**)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONSUMERINFO** = 5

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- bool **browser**
- **Pointer**< **ActiveMQDestination** > **destination**
- int **prefetchSize**
- int **maximumPendingMessageLimit**
- bool **dispatchAsync**
- std::string **selector**
- std::string **subscriptionName**
- bool **noLocal**
- bool **exclusive**
- bool **retroactive**
- unsigned char **priority**
- std::vector< **decaf::lang::Pointer**< **BrokerId** > > **brokerPath**
- **Pointer**< **BooleanExpression** > **additionalPredicate**
- bool **networkSubscription**
- bool **optimizedAcknowledge**
- bool **noRangeAcks**
- std::vector< **decaf::lang::Pointer**< **ConsumerId** > > **networkConsumerPath**

6.264.1 Constructor & Destructor Documentation

6.264.1.1 `activemq::commands::ConsumerInfo::ConsumerInfo ()`

6.264.1.2 `virtual activemq::commands::ConsumerInfo::~~ConsumerInfo ()`
[virtual]

6.264.2 Member Function Documentation

6.264.2.1 `virtual ConsumerInfo* activemq::commands::ConsumerInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.264.2.2 `virtual void activemq::commands::ConsumerInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.264.2.3 `Pointer<RemoveInfo> activemq::commands::ConsumerInfo::createRemoveCommand () const`

6.264.2.4 `virtual bool activemq::commands::ConsumerInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

- 6.264.2.5 `virtual const Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate () const [virtual]`
- 6.264.2.6 `virtual Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate () [virtual]`
- 6.264.2.7 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath () const [virtual]`
- 6.264.2.8 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath () [virtual]`
- 6.264.2.9 `virtual Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId () [virtual]`
- 6.264.2.10 `virtual const Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId () const [virtual]`
- 6.264.2.11 `virtual unsigned char activemq::commands::ConsumerInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.264.2.12 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination () const`
[virtual]
- 6.264.2.13 `virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination ()`
[virtual]
- 6.264.2.14 `virtual int activemq::commands::ConsumerInfo::getMaximumPendingMessageLimit () const` [virtual]
- 6.264.2.15 `virtual const std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath () const` [virtual]
- 6.264.2.16 `virtual std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath ()`
[virtual]
- 6.264.2.17 `virtual int activemq::commands::ConsumerInfo::getPrefetchSize () const` [virtual]
- 6.264.2.18 `virtual unsigned char activemq::commands::ConsumerInfo::getPriority () const` [virtual]
- 6.264.2.19 `virtual const std::string& activemq::commands::ConsumerInfo::getSelector () const` [virtual]
- 6.264.2.20 `virtual std::string& activemq::commands::ConsumerInfo::getSelector ()` [virtual]
- 6.264.2.21 `virtual const std::string& activemq::commands::ConsumerInfo::getSubscriptionName () const` [virtual]
- 6.264.2.22 `virtual std::string& activemq::commands::ConsumerInfo::getSubscriptionName ()` [virtual]
- 6.264.2.23 `virtual bool activemq::commands::ConsumerInfo::isBrowser () const` [virtual]
- 6.264.2.24 `virtual bool activemq::commands::ConsumerInfo::isConsumerInfo () const` [inline, virtual]

Returns

an answer of true to the `isConsumerInfo()` (p. 1362) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 698).

- 6.264.2.25 virtual bool activemq::commands::ConsumerInfo::isDispatchAsync () const [virtual]
- 6.264.2.26 virtual bool activemq::commands::ConsumerInfo::isExclusive () const [virtual]
- 6.264.2.27 virtual bool activemq::commands::ConsumerInfo::isNetworkSubscription () const [virtual]
- 6.264.2.28 virtual bool activemq::commands::ConsumerInfo::isNoLocal () const [virtual]
- 6.264.2.29 virtual bool activemq::commands::ConsumerInfo::isNoRangeAcks () const [virtual]
- 6.264.2.30 virtual bool activemq::commands::ConsumerInfo::isOptimizedAcknowledge () const [virtual]
- 6.264.2.31 virtual bool activemq::commands::ConsumerInfo::isRetroactive () const [virtual]
- 6.264.2.32 virtual void activemq::commands::ConsumerInfo::setAdditionalPredicate (const Pointer< BooleanExpression > & *additionalPredicate*) [virtual]
- 6.264.2.33 virtual void activemq::commands::ConsumerInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]
- 6.264.2.34 virtual void activemq::commands::ConsumerInfo::setBrowser (bool *browser*) [virtual]
- 6.264.2.35 virtual void activemq::commands::ConsumerInfo::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]
- 6.264.2.36 virtual void activemq::commands::ConsumerInfo::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.264.2.37 virtual void activemq::commands::ConsumerInfo::setDispatchAsync (bool *dispatchAsync*) [virtual]
- 6.264.2.38 virtual void activemq::commands::ConsumerInfo::setExclusive (bool *exclusive*) [virtual]
- 6.264.2.39 virtual void activemq::commands::ConsumerInfo::setMaximumPendingMessageLimit (int *maximumPendingMessageLimit*) [virtual]
- 6.264.2.40 virtual void activemq::commands::ConsumerInfo::setNetworkConsumerPath (const std::vector< decaf::lang::Pointer< ConsumerId > > & *networkConsumerPath*) [virtual]
- 6.264.2.41 virtual void activemq::commands::ConsumerInfo::setNetworkSubscription (bool *networkSubscription*) [virtual]
- 6.264.2.42 virtual void activemq::commands::ConsumerInfo::setNoLocal (bool *noLocal*) [virtual]
- 6.264.2.43 virtual void activemq::commands::ConsumerInfo::setNoRangeAcks (

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

6.264.2.51 `virtual Pointer<Command> activemq::commands::ConsumerInfo::visit
(activemq::state::CommandVisitor * visitor) throw (
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.264.3 Field Documentation

- 6.264.3.1 `Pointer<BooleanExpression> activemq::commands::ConsumerInfo::additionalPredicate` [protected]
- 6.264.3.2 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ConsumerInfo::brokerPath` [protected]
- 6.264.3.3 `bool activemq::commands::ConsumerInfo::browser` [protected]
- 6.264.3.4 `Pointer<ConsumerId> activemq::commands::ConsumerInfo::consumerId` [protected]
- 6.264.3.5 `Pointer<ActiveMQDestination> activemq::commands::ConsumerInfo::destination` [protected]
- 6.264.3.6 `bool activemq::commands::ConsumerInfo::dispatchAsync` [protected]
- 6.264.3.7 `bool activemq::commands::ConsumerInfo::exclusive` [protected]
- 6.264.3.8 `const unsigned char activemq::commands::ConsumerInfo::ID_ - CONSUMERINFO = 5` [static]
- 6.264.3.9 `int activemq::commands::ConsumerInfo::maximumPendingMessageLimit` [protected]
- 6.264.3.10 `std::vector< decaf::lang::Pointer<ConsumerId> > activemq::commands::ConsumerInfo::networkConsumerPath` [protected]
- 6.264.3.11 `bool activemq::commands::ConsumerInfo::networkSubscription` [protected]
- 6.264.3.12 `bool activemq::commands::ConsumerInfo::noLocal` [protected]
- 6.264.3.13 `bool activemq::commands::ConsumerInfo::noRangeAcks` [protected]
- 6.264.3.14 `bool activemq::commands::ConsumerInfo::optimizedAcknowledge` [protected]
- 6.264.3.15 `int activemq::commands::ConsumerInfo::prefetchSize` [protected]
- 6.264.3.16 `unsigned char activemq::commands::ConsumerInfo::priority` [protected]
- 6.264.3.17 `bool activemq::commands::ConsumerInfo::retroactive` [protected]
- 6.264.3.18 `std::string activemq::commands::ConsumerInfo::selector` [protected]
- 6.264.3.19 `std::string activemq::commands::ConsumerInfo::subscriptionName` [protected]

The documentation for this class was generated from the following file:

Generated on Sun Sep 11 2011 18:23:35 for activemq-cpp-3.2.1 by Doxygen

- `src/main/activemq/commands/ConsumerInfo.h`

6.265 `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerInfoMarshaller` (p. 1366).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller`:

Public Member Functions

- `ConsumerInfoMarshaller ()`
- `virtual ~ConsumerInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.265.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1366). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.265.2 Constructor & Destructor Documentation

6.265.2.1 `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::ConsumerInfoMarshaller () [inline]`

6.265.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller () [inline, virtual]`

6.265.3 Member Function Documentation

6.265.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.265.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.265.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.265.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

6.265.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

6.265.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

6.265.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h`

6.266 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for `ConsumerInfoMarshaller` (p. 1369).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller`:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.266.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1369). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.266.2 Constructor & Destructor Documentation

6.266.2.1 `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::ConsumerInfoMar
() [inline]`

6.266.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::~~ConsumerInfoMar
() [inline, virtual]`

6.266.3 Member Function Documentation

6.266.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.266.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.266.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.266.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.266.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

6.266.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 706).

```
6.266.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h`

6.267 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1373).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller**:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.267.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1373). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.267.2 Constructor & Destructor Documentation

6.267.2.1 `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::ConsumerInfoMar
() [inline]`

6.267.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::~~ConsumerInfoMar
() [inline, virtual]`

6.267.3 Member Function Documentation

6.267.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.267.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.267.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`
(p. 709).

6.267.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

6.267.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

6.267.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.267.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h`

6.268 `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerInfoMarshaller` (p. 1377).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller`:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.268.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1377). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.268.2 Constructor & Destructor Documentation

6.268.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::ConsumerInfoMar
() [inline]`

6.268.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::~~ConsumerInfoMar
() [inline, virtual]`

6.268.3 Member Function Documentation

6.268.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.268.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.268.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`
(p. 716).

6.268.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.268.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

6.268.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

```
6.268.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h`

6.269 `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerInfoMarshaller` (p. 1381).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller`:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.269.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1381). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.269.2 Constructor & Destructor Documentation

6.269.2.1 `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::ConsumerInfoMarshaller() [inline]`

6.269.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller() [inline, virtual]`

6.269.3 Member Function Documentation

6.269.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::createObject() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.269.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.269.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.269.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

6.269.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

6.269.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.269.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h`

6.270 `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerInfoMarshaller` (p. 1385).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller`:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.270.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1385). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.270.2 Constructor & Destructor Documentation

6.270.2.1 `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::ConsumerInfoMar
() [inline]`

6.270.2.2 `virtual
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::~~ConsumerInfoMar
() [inline, virtual]`

6.270.3 Member Function Documentation

6.270.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.270.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.270.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller`
(p. 729).

6.270.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

6.270.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

6.270.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 732).

```
6.270.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h

6.271 activemq::state::ConsumerState Class Reference

```
#include <src/main/activemq/state/ConsumerState.h>
```

Public Member Functions

- **ConsumerState** (const **Pointer**< **ConsumerInfo** > &info)
- virtual ~**ConsumerState** ()
- std::string **toString** () const
- const **Pointer**< **ConsumerInfo** > & **getInfo** () const

6.271.1 Constructor & Destructor Documentation

6.271.1.1 `activemq::state::ConsumerState::ConsumerState (const Pointer< ConsumerInfo > & info)`

6.271.1.2 `virtual activemq::state::ConsumerState::~~ConsumerState () [virtual]`

6.271.2 Member Function Documentation

6.271.2.1 `const Pointer<ConsumerInfo>& activemq::state::ConsumerState::getInfo () const [inline]`

6.271.2.2 `std::string activemq::state::ConsumerState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConsumerState.h`

6.272 activemq::commands::ControlCommand Class Reference

```
#include <src/main/activemq/commands/ControlCommand.h>
```

Inheritance diagram for `activemq::commands::ControlCommand`:

Public Member Functions

- `ControlCommand ()`
- `virtual ~ControlCommand ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual ControlCommand * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- `virtual void copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this object's members, overwriting any existing data.
- `virtual std::string toString () const`
Returns a string containing the information for this DataStructure (p. 1553) such as its type and value of its elements.
- `virtual bool equals (const DataStructure *value) const`
Compares the DataStructure (p. 1553) passed in to this one, and returns if they are equivalent.
- `virtual const std::string & getCommand () const`
- `virtual std::string & getCommand ()`

- virtual void **setCommand** (const std::string &command)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONTROLCOMMAND** = 14

Protected Attributes

- std::string **command**

6.272.1 Constructor & Destructor Documentation

6.272.1.1 **activemq::commands::ControlCommand::ControlCommand** ()

6.272.1.2 **virtual activemq::commands::ControlCommand::~~ControlCommand** ()
[virtual]

6.272.2 Member Function Documentation

6.272.2.1 **virtual ControlCommand* activemq::commands::ControlCommand::cloneDataStructure**
() const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.272.2.2 **virtual void activemq::commands::ControlCommand::copyDataStructure**
(const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.272.2.3 `virtual bool activemq::commands::ControlCommand::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.272.2.4 `virtual std::string& activemq::commands::ControlCommand::getCommand ()` [virtual]

6.272.2.5 `virtual const std::string& activemq::commands::ControlCommand::getCommand () const` [virtual]

6.272.2.6 `virtual unsigned char activemq::commands::ControlCommand::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

6.272.2.7 `virtual void activemq::commands::ControlCommand::setCommand (const std::string & command)` [virtual]

6.272.2.8 `virtual std::string activemq::commands::ControlCommand::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

6.272.2.9 virtual Pointer<Command> activemq::commands::ControlCommand::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.272.3 Field Documentation

6.272.3.1 std::string activemq::commands::ControlCommand::command [protected]

6.272.3.2 const unsigned char activemq::commands::ControlCommand::ID _ - CONTROLCOMMAND = 14 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/ControlCommand.h

6.273 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1393).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual ~**ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.273.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p.1393). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.273.2 Constructor & Destructor Documentation

6.273.2.1 **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::ControlComm**
() [inline]

6.273.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::~~ControlComm
() [inline, virtual]

6.273.3 Member Function Documentation

6.273.3.1 **virtual commands::DataStructure* ac-**
tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.273.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.273.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 736).

6.273.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 737).

6.273.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

6.273.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

6.273.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h`

6.274 `activemq::wireformat::openwire::marshal::v3::ControlCommandM` Class Reference

Marshaling code for Open Wire Format for `ControlCommandMarshaller` (p. 1397).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller`:

Public Member Functions

- `ControlCommandMarshaller ()`
- `virtual ~ControlCommandMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.274.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1397). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.274.2 Constructor & Destructor Documentation

6.274.2.1 **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::ControlComm**
() [inline]

6.274.2.2 **virtual**
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::~~ControlComm
() [inline, virtual]

6.274.3 Member Function Documentation

6.274.3.1 **virtual commands::DataStructure* ac-**
tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.274.3.2 **virtual unsigned char ac-**
tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::getDataStructur
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.274.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.274.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.274.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

```
6.274.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.274.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - `BooleanStream` stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h`

6.275 `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `ControlCommandMarshaller` (p. 1401).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller`:

Public Member Functions

- `ControlCommandMarshaller ()`
- `virtual ~ControlCommandMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.275.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1401). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.275.2 Constructor & Destructor Documentation

6.275.2.1 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::ControlCommandMarshaller () [inline]

6.275.2.2 virtual activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::~~ControlCommandMarshaller () [inline, virtual]

6.275.3 Member Function Documentation

6.275.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.275.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.275.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 709).

6.275.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 710).

6.275.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

```
6.275.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.275.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h`

6.276 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1405).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller**:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.276.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1405). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.276.2 Constructor & Destructor Documentation

6.276.2.1 `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::ControlCommandMarshaller () [inline]`

6.276.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::~~ControlCommandMarshaller () [inline, virtual]`

6.276.3 Member Function Documentation

6.276.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.276.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.276.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 716).

6.276.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 717).

6.276.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

```
6.276.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

```
6.276.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException*** if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ControlCommandMarshaller.h**

6.277 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1409).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller**:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.277.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1409). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.277.2 Constructor & Destructor Documentation

6.277.2.1 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::ControlCommandMarshaller () [inline]

6.277.2.2 virtual
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::~~ControlCommandMarshaller () [inline, virtual]

6.277.3 Member Function Documentation

6.277.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.277.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.277.3.3 virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 722).

6.277.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 723).

6.277.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```
6.277.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.277.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h`

6.278 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1413).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller**:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.278.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1413). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.278.2 Constructor & Destructor Documentation

6.278.2.1 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::ControlCommandMarshaller () [inline]

6.278.2.2 virtual
activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::~~ControlCommandMarshaller () [inline, virtual]

6.278.3 Member Function Documentation

6.278.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.278.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.278.3.3 virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 729).

6.278.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 730).

6.278.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

```
6.278.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

```
6.278.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller.h`

6.279 decaf::util::concurrent::CountDownLatch Class Reference

```
#include <src/main/decaf/util/concurrent/CountDownLatch.h>
```

Public Member Functions

- **CountDownLatch** (int count)
Constructor.
- virtual **~CountDownLatch** ()
- virtual void **await** () throw (decaf::lang::exceptions::InterruptedException, decaf::lang::Exception)
Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted.
- virtual bool **await** (long long timeOut) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::Exception)
Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.
- virtual bool **await** (long long timeout, const **TimeUnit** &unit) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::Exception)
Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.
- virtual void **countDown** ()
Counts down the latch, releasing all waiting threads when the count hits zero.
- virtual int **getCount** () const
Gets the current count.

6.279.1 Constructor & Destructor Documentation

6.279.1.1 decaf::util::concurrent::CountDownLatch::CountDownLatch (int count)

Constructor.

Parameters

count - number to count down from.

6.279.1.2 `virtual decaf::util::concurrent::CountDownLatch::~~CountDownLatch () [virtual]`

6.279.2 Member Function Documentation

6.279.2.1 `virtual void decaf::util::concurrent::CountDownLatch::await () throw (decaf::lang::exceptions::InterruptedException, decaf::lang::Exception) [virtual]`

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted.

If the current count is zero then this method returns immediately.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happen:

* The count reaches zero due to invocations of the **countDown()** (p.1419) method; or * Some other thread interrupts the current thread.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting, then InterruptedException is thrown and the current thread's interrupted status is cleared.

Exceptions

InterruptedException - if the current thread is interrupted while waiting.

Exception - if any other error occurs.

6.279.2.2 `virtual bool decaf::util::concurrent::CountDownLatch::await (long long timeOut) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::Exception) [virtual]`

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

If the current count is zero then this method returns immediately with the value true.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happen:

* The count reaches zero due to invocations of the **countDown()** (p.1419) method; or * Some other thread interrupts the current thread; or * The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting, then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Parameters

timeout - Time in milliseconds to wait for the count to reach zero.

Exceptions

InterruptedException - if the current thread is interrupted while waiting.

Exception - if any other error occurs.

6.279.2.3 `virtual bool decaf::util::concurrent::CountDownLatch::await
(long long timeout, const TimeUnit & unit) throw (
decaf::lang::exceptions::InterruptedException, decaf::lang::Exception)
[virtual]`

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

If the current count is zero then this method returns immediately with the value true.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happen:

* The count reaches zero due to invocations of the **countDown()** (p. 1419) method; or * Some other thread interrupts the current thread; or * The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Parameters

timeout - Time to wait for the count to reach zero.

unit - The units that the timeout specifies.

Exceptions

InterruptedException - if the current thread is interrupted while waiting.

Exception - if any other error occurs.

6.279.2.4 `virtual void decaf::util::concurrent::CountDownLatch::countDown ()
[virtual]`

Counts down the latch, releasing all waiting threads when the count hits zero.

6.279.2.5 `virtual int decaf::util::concurrent::CountDownLatch::getCount () const
[inline, virtual]`

Gets the current count.

Returns

int count value

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/CountDownLatch.h`

6.280 decaf::util::zip::CRC32 Class Reference

Class that can be used to compute a CRC-32 checksum for a data stream.

```
#include <src/main/decaf/util/zip/CRC32.h>
```

Inheritance diagram for `decaf::util::zip::CRC32`:

Public Member Functions

- **CRC32** ()
- virtual **~CRC32** ()
- virtual long long **getValue** () const
- virtual void **reset** ()
Reset the checksum to its initial value.
- virtual void **update** (const std::vector< unsigned char > &buffer)
Updates the current checksum with the specified vector of bytes.
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Updates the current checksum with the specified array of bytes.
- virtual void **update** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Updates the current checksum with the specified array of bytes.
- virtual void **update** (int byte)
Updates the current checksum with the specified byte value.

6.280.1 Detailed Description

Class that can be used to compute a CRC-32 checksum for a data stream.

Since

1.0

6.280.2 Constructor & Destructor Documentation

6.280.2.1 decaf::util::zip::CRC32::CRC32 ()

6.280.2.2 virtual decaf::util::zip::CRC32::~~CRC32 () [virtual]

6.280.3 Member Function Documentation

6.280.3.1 virtual long long decaf::util::zip::CRC32::getValue () const [virtual]

Returns

the current checksum value.

Implements **decaf::util::zip::Checksum** (p. 1060).

6.280.3.2 virtual void decaf::util::zip::CRC32::reset () [virtual]

Reset the checksum to its initial value.

Implements **decaf::util::zip::Checksum** (p. 1060).

6.280.3.3 virtual void decaf::util::zip::CRC32::update (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer The buffer to read the updated bytes from.

size The size of the passed buffer.

offset The position in the buffer to start reading.

length The amount of data to read from the byte buffer.

Exceptions

NullPointerException if the passed buffer is NULL.

IndexOutOfBoundsException if offset + length > size of the buffer.

Implements **decaf::util::zip::Checksum** (p. 1061).

6.280.3.4 virtual void decaf::util::zip::CRC32::update (const std::vector< unsigned char > & *buffer*, int *offset*, int *length*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer The buffer to read the updated bytes from.

offset The position in the buffer to start reading.

length The amount of data to read from the byte buffer.

Exceptions

IndexOutOfBoundsException if $\text{offset} + \text{length} > \text{size of the buffer}$.

Implements **decaf::util::zip::Checksum** (p. 1061).

6.280.3.5 `virtual void decaf::util::zip::CRC32::update (const std::vector< unsigned char > & buffer) [virtual]`

Updates the current checksum with the specified vector of bytes.

Parameters

buffer The buffer to read the updated bytes from.

Implements **decaf::util::zip::Checksum** (p. 1061).

6.280.3.6 `virtual void decaf::util::zip::CRC32::update (int byte) [virtual]`

Updates the current checksum with the specified byte value.

Parameters

byte The byte value to update the current **Checksum** (p. 1059) with (0..255).

Implements **decaf::util::zip::Checksum** (p. 1062).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/CRC32.h`

6.281 ct_data_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

Data Fields

- union {
 ush freq
 ush code
} fc
- union {
 ush dad
 ush len
} dl

6.281.1 Field Documentation

6.281.1.1 `ush ct_data_s::code`

6.281.1.2 `ush ct_data_s::dad`

6.281.1.3 `union { ... } ct_data_s::dl`

6.281.1.4 `union { ... } ct_data_s::fc`

6.281.1.5 `ush ct_data_s::freq`

6.281.1.6 `ush ct_data_s::len`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/deflate.h`

6.282 activemq::commands::DataArrayResponse Class Reference

```
#include <src/main/activemq/commands/DataArrayResponse.h>
```

Inheritance diagram for `activemq::commands::DataArrayResponse`:

Public Member Functions

- **DataArrayResponse** ()
- virtual **~DataArrayResponse** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DataArrayResponse * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const std::vector< **decaf::lang::Pointer**< **DataStructure** > & **getData** () const
- virtual std::vector< **decaf::lang::Pointer**< **DataStructure** > & **getData** ()

- virtual void **setData** (const std::vector< **decaf::lang::Pointer**< **DataStructure** > > &**data**)

Static Public Attributes

- static const unsigned char **ID_DATAARRAYRESPONSE** = 33

Protected Attributes

- std::vector< **decaf::lang::Pointer**< **DataStructure** > > **data**

6.282.1 Constructor & Destructor Documentation

6.282.1.1 **activemq::commands::DataArrayResponse::DataArrayResponse** ()

6.282.1.2 **virtual activemq::commands::DataArrayResponse::~~DataArrayResponse** () [virtual]

6.282.2 Member Function Documentation

6.282.2.1 **virtual DataArrayResponse* activemq::commands::DataArrayResponse::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::Response** (p.3077).

6.282.2.2 **virtual void activemq::commands::DataArrayResponse::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::Response** (p.3078).

6.282.2.3 **virtual bool activemq::commands::DataArrayResponse::equals** (const **DataStructure** * *value*) const [virtual]

Compares the **DataStructure** (p.1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Response** (p.3078).

6.282.2.4 `virtual std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData () [virtual]`

6.282.2.5 `virtual const std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData () const [virtual]`

6.282.2.6 `virtual unsigned char activemq::commands::DataArrayResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p.1553) type copy.

Reimplemented from **activemq::commands::Response** (p.3078).

6.282.2.7 `virtual void activemq::commands::DataArrayResponse::setData (const std::vector< decaf::lang::Pointer< DataStructure > > & data) [virtual]`

6.282.2.8 `virtual std::string activemq::commands::DataArrayResponse::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p.3079).

6.282.3 Field Documentation

6.282.3.1 `std::vector< decaf::lang::Pointer<DataStructure> > activemq::commands::DataArrayResponse::data [protected]`

6.282.3.2 `const unsigned char activemq::commands::DataArrayResponse::ID_ - DATAARRAYRESPONSE = 33 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataArrayResponse.h`

6.283 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1426).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller`:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.283.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1426). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.283.2 Constructor & Destructor Documentation

6.283.2.1 `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::DataArrayR
() [inline]`

6.283.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::~~DataArrayR
() [inline, virtual]`

6.283.3 Member Function Documentation

6.283.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3090).

6.283.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::getDataStruct
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3091).

6.283.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3091).

```
6.283.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3091).

```
6.283.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3092).

6.283.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3092).

6.283.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3093).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h`

6.284 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1430).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller`:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.284.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1430). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.284.2 Constructor & Destructor Documentation

6.284.2.1 `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::DataArrayR
() [inline]`

6.284.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::~~DataArrayR
() [inline, virtual]`

6.284.3 Member Function Documentation

6.284.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3099).

6.284.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::getDataStruct
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3100).

6.284.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3100).

```
6.284.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3100).

```
6.284.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3101).

6.284.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3101).

6.284.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3102).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h`

6.285 activemq::wireformat::openwire::marshal::v4::DataArrayResponse Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1434).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.285.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1434). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.285.2 Constructor & Destructor Documentation

6.285.2.1 `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::DataArrayR
() [inline]`

6.285.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::~~DataArrayR
() [inline, virtual]`

6.285.3 Member Function Documentation

6.285.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3086).

6.285.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::getDataStruct
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3086).

6.285.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3086).

```
6.285.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3087).

```
6.285.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3087).

6.285.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3088).

6.285.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3088).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h`

6.286 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1438).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.286.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1438). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.286.2 Constructor & Destructor Documentation

6.286.2.1 `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::DataArrayR
() [inline]`

6.286.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::~~DataArrayR
() [inline, virtual]`

6.286.3 Member Function Documentation

6.286.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3104).

6.286.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::getDataStruct
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3104).

6.286.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3104).

```
6.286.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3105).

```
6.286.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3105).

6.286.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3106).

6.286.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3106).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h`

6.287 activemq::wireformat::openwire::marshal::v5::DataArrayResponse Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1442).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.287.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1442). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.287.2 Constructor & Destructor Documentation

6.287.2.1 `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::DataArrayR
() [inline]`

6.287.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::~~DataArrayR
() [inline, virtual]`

6.287.3 Member Function Documentation

6.287.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3095).

6.287.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::getDataStruct
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3095).

6.287.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3095).

```
6.287.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3096).

```
6.287.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3096).

6.287.3.6 virtual void `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal2`
(`OpenWireFormat` * *wireFormat*, `commands::DataStructure`
* *dataStructure*, `decaf::io::DataOutputStream` * *dataOut*,
`utils::BooleanStream` * *bs*) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3097).

6.287.3.7 virtual void `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightUnmarshal`
(`OpenWireFormat` * *wireFormat*, `commands::DataStructure`
* *dataStructure*, `decaf::io::DataInputStream` * *dataIn*,
`utils::BooleanStream` * *bs*) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3097).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h`

6.288 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1446).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.288.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1446). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.288.2 Constructor & Destructor Documentation

6.288.2.1 `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::DataArrayR
() [inline]`

6.288.2.2 `virtual
activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::~~DataArrayR
() [inline, virtual]`

6.288.3 Member Function Documentation

6.288.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3108).

6.288.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::getDataStruct
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109).

6.288.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109).

```
6.288.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109).

```
6.288.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3110).

6.288.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3110).

6.288.3.7 `virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3111).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h`

6.289 decaf::util::zip::DataFormatException Class Reference

```
#include <src/main/decaf/util/zip/DataFormatException.h>
```

Inheritance diagram for decaf::util::zip::DataFormatException:

Public Member Functions

- **DataFormatException** () throw ()
Default Constructor.
- **DataFormatException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **DataFormatException** (const DataFormatException &ex) throw ()
Copy Constructor.
- **DataFormatException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **DataFormatException** (const std::exception *cause) throw ()
Constructor.
- **DataFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **DataFormatException** * clone () const
Clones this exception.
- virtual ~**DataFormatException** () throw ()

6.289.1 Constructor & Destructor Documentation

6.289.1.1 decaf::util::zip::DataFormatException::DataFormatException () throw () [inline]

Default Constructor.

6.289.1.2 decaf::util::zip::DataFormatException::DataFormatException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.289.1.3 decaf::util::zip::DataFormatException::DataFormatException (const DataFormatException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.289.1.4 decaf::util::zip::DataFormatException::DataFormatException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.289.1.5 decaf::util::zip::DataFormatException::DataFormatException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.289.1.6 decaf::util::zip::DataFormatException::DataFormatException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.289.1.7 `virtual decaf::util::zip::DataFormatException::~DataFormatException () throw () [inline, virtual]`

6.289.2 Member Function Documentation

6.289.2.1 `virtual DataFormatException* decaf::util::zip::DataFormatException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an Exception that is a copy of this instance.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/DataFormatException.h`

6.290 decaf::io::DataInput Class Reference

The **DataInput** (p. 1452) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.

```
#include <src/main/decaf/io/DataInput.h>
```

Public Member Functions

- `virtual ~DataInput ()`
- `virtual bool readBoolean ()=0 throw (decaf::io::IOException, decaf::io::EOFException)`
Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.
- `virtual char readByte ()=0 throw (decaf::io::IOException, decaf::io::EOFException)`
Reads and returns one input byte.
- `virtual unsigned char readUnsignedByte ()=0 throw (decaf::io::IOException, decaf::io::EOFException)`
Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.
- `virtual char readChar ()=0 throw (decaf::io::IOException, decaf::io::EOFException)`
Reads an input char and returns the char value.
- `virtual double readDouble ()=0 throw (decaf::io::IOException, decaf::io::EOFException)`
Reads eight input bytes and returns a double value.

- virtual float **readFloat** ()=0 throw (decaf::io::IOException, decaf::io::EOFException)
Reads four input bytes and returns a float value.
- virtual int **readInt** ()=0 throw (decaf::io::IOException, decaf::io::EOFException)
Reads four input bytes and returns an int value.
- virtual long long **readLong** ()=0 throw (decaf::io::IOException, decaf::io::EOFException)
Reads eight input bytes and returns a long value.
- virtual short **readShort** ()=0 throw (decaf::io::IOException, decaf::io::EOFException)
Reads two input bytes and returns a short value.
- virtual unsigned short **readUnsignedShort** ()=0 throw (decaf::io::IOException, decaf::io::EOFException)
Reads two input bytes and returns an int value in the range 0 through 65535.
- virtual std::string **readString** ()=0 throw (decaf::io::IOException, decaf::io::EOFException)
Reads an NULL terminated ASCII string to the stream and returns the string to the caller.
- virtual std::string **readLine** ()=0 throw (decaf::io::IOException)
Reads the next line of text from the input stream.
- virtual std::string **readUTF** ()=0 throw (decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException)
Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.
- virtual void **readFully** (unsigned char *buffer, int size)=0 throw (decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException)
Reads some bytes from an input stream and stores them into the buffer array buffer.
- virtual void **readFully** (unsigned char *buffer, int size, int offset, int length)=0 throw (decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Reads length bytes from an input stream.
- virtual long long **skipBytes** (long long num)=0 throw (io::IOException)
Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

6.290.1 Detailed Description

The **DataInput** (p. 1452) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types. There is also a facility for reconstructing Strings from data in the Java standard modified UTF-8 format.

It is generally true of all the reading routines in this interface that if end of file is reached before the desired number of bytes has been read, an **EOFException** (p. 1707) is thrown. If any byte

cannot be read for any reason other than end of file, an **IOException** (p. 2003) other than **EOFException** (p. 1707) is thrown. for example, an **IOException** (p. 2003) may be thrown if the underlying input stream has been closed.

See also

DataOutput (p. 1468)

DataInputStream (p. 1460)

Since

1.0

6.290.2 Constructor & Destructor Documentation

6.290.2.1 `virtual decaf::io::DataInput::~~DataInput () [inline, virtual]`

6.290.3 Member Function Documentation

6.290.3.1 `virtual bool decaf::io::DataInput::readBoolean () throw (decaf::io::IOException, decaf::io::EOFException) [pure virtual]`

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

Returns

the boolean value of the read in byte (0=false, 1=true).

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.290.3.2 `virtual char decaf::io::DataInput::readByte () throw (decaf::io::IOException, decaf::io::EOFException) [pure virtual]`

Reads and returns one input byte.

The byte is treated as a signed value in the range -128 through 127, inclusive.

Returns

the 8-bit value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.290.3.3 virtual char decaf::io::DataInput::readChar () throw (decaf::io::IOException, decaf::io::EOFException) [pure virtual]

Reads an input char and returns the char value.

A ascii char is made up of one bytes. This returns the same result as `readByte`

Returns

the 8 bit char read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.290.3.4 virtual double decaf::io::DataInput::readDouble () throw (decaf::io::IOException, decaf::io::EOFException) [pure virtual]

Reads eight input bytes and returns a double value.

It does this by first constructing a long long value in exactly the manner of the `readlong` method, then converting this long value to a double in exactly the manner of the method `Double::longBitsToDouble`.

Returns

the double value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.290.3.5 virtual float decaf::io::DataInput::readFloat () throw (decaf::io::IOException, decaf::io::EOFException) [pure virtual]

Reads four input bytes and returns a float value.

It does this by first constructing an int value in exactly the manner of the `readInt` method, then converting this int value to a float in exactly the manner of the method `Float::intBitsToFloat`.

Returns

the float value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.290.3.6 virtual void decaf::io::DataInput::readFully (unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Reads length bytes from an input stream.

This method blocks until one of the following conditions occurs: * length bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1707) is thrown. * An I/O error occurs, in which case an **IOException** (p. 2003) other than **EOFException** (p. 1707) is thrown.

If buffer is NULL, a NullPointerException is thrown. If offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If length is zero, then no bytes are read. Otherwise, the first byte read is stored into element buffer[off], the next one into buffer[offset+1], and so on. The number of bytes read is, at most, equal to length.

Parameters

- buffer* The byte array to insert read data into.
- size* The size in bytes of the given byte buffer.
- offset* The location in buffer to start writing.
- length* The number of bytes to read from the buffer.

Exceptions

- IOException** (p. 2003) if an I/O Error occurs.
- EOFException** (p. 1707) if the end of input is reached.
- NullPointerException** if the buffer is NULL.
- IndexOutOfBoundsException** if the offset + length > size, or an int param is negative.

6.290.3.7 virtual void decaf::io::DataInput::readFully (unsigned char * *buffer*, int *size*) throw (decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Reads some bytes from an input stream and stores them into the buffer array buffer.

The number of bytes read is equal to the length of buffer.

This method blocks until one of the following conditions occurs: * buffer's size bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1707) is thrown. * An I/O error occurs, in which case an **IOException** (p. 2003) other than **EOFException** (p. 1707) is thrown.

If buffer size is zero, then no bytes are read. Otherwise, the first byte read is stored into element b[0], the next one into buffer[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of buffer have been updated with data from the input stream.

Parameters

- buffer* The byte array to insert read data into.
- size* The size in bytes of the given byte buffer.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

IndexOutOfBoundsException if the size value is negative.

6.290.3.8 virtual int decaf::io::DataInput::readInt () throw (decaf::io::IOException, decaf::io::EOFException) [pure virtual]

Reads four input bytes and returns an int value.

Let a be the first byte read, b be the second byte, c be the third byte, and d be the fourth byte. The value returned is:

$$(((a \& 0xff) << 24) | ((b \& 0xff) << 16) | ((c \& 0xff) << 8) | (d \& 0xff))$$

Returns

the int value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.290.3.9 virtual std::string decaf::io::DataInput::readLine () throw (decaf::io::IOException) [pure virtual]

Reads the next line of text from the input stream.

It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character ' ' is encountered, it is discarded and reading ceases. If the character " is encountered, it is discarded and, if the following byte converts to the character ' ' , then that is discarded also; reading then ceases. If end of file is encountered before either of the characters ' ' and " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

Returns

the next line of text read from the input stream or empty string if at EOF.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

6.290.3.10 `virtual long long decaf::io::DataInput::readLong () throw (decaf::io::IOException, decaf::io::EOFException) [pure virtual]`

Reads eight input bytes and returns a long value.

Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth byte. The value returned is:

$$(((\text{long})(a \ \& \ 0\text{xff}) \ll 56) \mid ((\text{long})(b \ \& \ 0\text{xff}) \ll 48) \mid ((\text{long})(c \ \& \ 0\text{xff}) \ll 40) \mid ((\text{long})(d \ \& \ 0\text{xff}) \ll 32) \mid ((\text{long})(e \ \& \ 0\text{xff}) \ll 24) \mid ((\text{long})(f \ \& \ 0\text{xff}) \ll 16) \mid ((\text{long})(g \ \& \ 0\text{xff}) \ll 8) \mid ((\text{long})(h \ \& \ 0\text{xff})))$$

Returns

the 64 bit long long read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.290.3.11 `virtual short decaf::io::DataInput::readShort () throw (decaf::io::IOException, decaf::io::EOFException) [pure virtual]`

Reads two input bytes and returns a short value.

Let a be the first byte read and b be the second byte. The value returned is:

$$(\text{short})((a \ll 8) \mid (b \ \& \ 0\text{xff}))$$

Returns

the 16 bit short value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.290.3.12 `virtual std::string decaf::io::DataInput::readString () throw (decaf::io::IOException, decaf::io::EOFException) [pure virtual]`

Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

Returns

string object containing the string read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.290.3.13 `virtual unsigned char decaf::io::DataInput::readUnsignedByte ()
 throw (decaf::io::IOException, decaf::io::EOFException) [pure
 virtual]`

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

Returns

the 8 bit unsigned value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.290.3.14 `virtual unsigned short decaf::io::DataInput::readUnsignedShort ()
 throw (decaf::io::IOException, decaf::io::EOFException) [pure
 virtual]`

Reads two input bytes and returns an int value in the range 0 through 65535.

Let a be the first byte read and b be the second byte. The value returned is:

$((a \& 0xff) \ll 8) \mid (b \& 0xff)$

Returns

the 16 bit unsigned short read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.290.3.15 `virtual std::string decaf::io::DataInput::readUTF ()
 throw (decaf::io::IOException, decaf::io::EOFException,
 decaf::io::UTFDataFormatException) [pure virtual]`

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.

This method reads String value written from a Java **DataOutputStream** (p. 1473) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

Returns

The decoded string read from stream.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

UTFDataFormatException (p. 3703) if the bytes are not valid modified UTF-8 values.

6.290.3.16 `virtual long long decaf::io::DataInput::skipBytes (long long num)
throw (io::IOException)` [pure virtual]

Makes an attempt to skip over *n* bytes of data from the input stream, discarding the skipped bytes.

However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. This method never throws an **EOFException** (p.1707). The actual number of bytes skipped is returned.

Parameters

num The number of bytes to skip over.

Returns

the total number of bytes skipped.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataInput.h`

6.291 decaf::io::DataInputStream Class Reference

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

```
#include <src/main/decaf/io/DataInputStream.h>
```

Inheritance diagram for decaf::io::DataInputStream:

Public Member Functions

- **DataInputStream** (**InputStream** *inputStream, bool own=false)
*Creates a **DataInputStream** (p.1460) that uses the specified underlying **InputStream** (p.1909).*
- virtual ~**DataInputStream** ()
- virtual bool **readBoolean** () throw (decaf::io::IOException, decaf::io::EOFException)
Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.
- virtual char **readByte** () throw (decaf::io::IOException, decaf::io::EOFException)
Reads and returns one input byte.
- virtual unsigned char **readUnsignedByte** () throw (decaf::io::IOException, decaf::io::EOFException)

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

- virtual char **readChar** () throw (decaf::io::IOException, decaf::io::EOFException)
Reads an input char and returns the char value.
- virtual double **readDouble** () throw (decaf::io::IOException, decaf::io::EOFException)
Reads eight input bytes and returns a double value.
- virtual float **readFloat** () throw (decaf::io::IOException, decaf::io::EOFException)
Reads four input bytes and returns a float value.
- virtual int **readInt** () throw (decaf::io::IOException, decaf::io::EOFException)
Reads four input bytes and returns an int value.
- virtual long long **readLong** () throw (decaf::io::IOException, decaf::io::EOFException)
Reads eight input bytes and returns a long value.
- virtual short **readShort** () throw (decaf::io::IOException, decaf::io::EOFException)
Reads two input bytes and returns a short value.
- virtual unsigned short **readUnsignedShort** () throw (decaf::io::IOException, decaf::io::EOFException)
Reads two input bytes and returns an int value in the range 0 through 65535.
- virtual std::string **readString** () throw (decaf::io::IOException, decaf::io::EOFException)
Reads an NULL terminated ASCII string to the stream and returns the string to the caller.
- virtual std::string **readLine** () throw (decaf::io::IOException)
Reads the next line of text from the input stream.
- virtual std::string **readUTF** () throw (decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException)
Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.
- virtual void **readFully** (unsigned char *buffer, int size) throw (decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException)
Reads some bytes from an input stream and stores them into the buffer array buffer.
- virtual void **readFully** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Reads length bytes from an input stream.
- virtual long long **skipBytes** (long long num) throw (io::IOException)
Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

6.291.1 Detailed Description

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way. An application uses a data output stream to write data that can later be read by a data input stream.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped `InputStream` (p. 1909) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

```
DataInputStream (p. 1460) os = new DataInputStream (p. 1460)( new InputStream(
(p. 1911), true )
```

Since

1.0

6.291.2 Constructor & Destructor Documentation

6.291.2.1 `decaf::io::DataInputStream::DataInputStream (InputStream * inputStream, bool own = false)`

Creates a `DataInputStream` (p. 1460) that uses the specified underlying `InputStream` (p. 1909).

Parameters

inputStream the `InputStream` (p. 1909) instance to wrap.

own indicates if this class owns the wrapped string defaults to false.

6.291.2.2 `virtual decaf::io::DataInputStream::~~DataInputStream ()` [virtual]

6.291.3 Member Function Documentation

6.291.3.1 `virtual bool decaf::io::DataInputStream::readBoolean ()` throw (`decaf::io::IOException`, `decaf::io::EOFException`) [virtual]

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

Returns

the boolean value of the read in byte (0=false, 1=true).

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.291.3.2 `virtual char decaf::io::DataInputStream::readByte ()` throw (`decaf::io::IOException`, `decaf::io::EOFException`) [virtual]

Reads and returns one input byte.

The byte is treated as a signed value in the range -128 through 127, inclusive.

Returns

the 8-bit value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.291.3.3 virtual char decaf::io::DataInputStream::readChar () throw (decaf::io::IOException, decaf::io::EOFException) [virtual]

Reads an input char and returns the char value.

A ascii char is made up of one bytes. This returns the same result as `readByte`

Returns

the 8 bit char read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.291.3.4 virtual double decaf::io::DataInputStream::readDouble () throw (decaf::io::IOException, decaf::io::EOFException) [virtual]

Reads eight input bytes and returns a double value.

It does this by first constructing a long long value in exactly the manner of the `readlong` method, then converting this long value to a double in exactly the manner of the method `Double::longBitsToDouble`.

Returns

the double value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.291.3.5 virtual float decaf::io::DataInputStream::readFloat () throw (decaf::io::IOException, decaf::io::EOFException) [virtual]

Reads four input bytes and returns a float value.

It does this by first constructing an int value in exactly the manner of the `readInt` method, then converting this int value to a float in exactly the manner of the method `Float::intBitsToFloat`.

Returns

the float value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.291.3.6 virtual void decaf::io::DataInputStream::readFully (unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads length bytes from an input stream.

This method blocks until one of the following conditions occurs: * length bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1707) is thrown. * An I/O error occurs, in which case an **IOException** (p. 2003) other than **EOFException** (p. 1707) is thrown.

If buffer is NULL, a NullPointerException is thrown. If offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If length is zero, then no bytes are read. Otherwise, the first byte read is stored into element buffer[offset], the next one into buffer[offset+1], and so on. The number of bytes read is, at most, equal to length.

Parameters

buffer The byte array to insert read data into.

size The size in bytes of the given byte buffer.

offset The location in buffer to start writing.

length The number of bytes to read from the buffer.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

NullPointerException if the buffer is NULL.

IndexOutOfBoundsException if the offset + length > size.

6.291.3.7 virtual void decaf::io::DataInputStream::readFully (unsigned char * *buffer*, int *size*) throw (decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads some bytes from an input stream and stores them into the buffer array buffer.

The number of bytes read is equal to the length of buffer.

This method blocks until one of the following conditions occurs: * buffer's size bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1707) is thrown. * An I/O error occurs, in which case an **IOException** (p. 2003) other than **EOFException** (p. 1707) is thrown.

If buffer size is zero, then no bytes are read. Otherwise, the first byte read is stored into element b[0], the next one into buffer[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of buffer have been updated with data from the input stream.

Parameters

buffer The byte array to insert read data into.

size The size in bytes of the given byte buffer.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

IndexOutOfBoundsException if the size value is negative.

6.291.3.8 `virtual int decaf::io::DataInputStream::readInt () throw (decaf::io::IOException, decaf::io::EOFException) [virtual]`

Reads four input bytes and returns an int value.

Let a be the first byte read, b be the second byte, c be the third byte, and d be the fourth byte. The value returned is:

$$(((a \& 0xff) << 24) | ((b \& 0xff) << 16) | ((c \& 0xff) << 8) | (d \& 0xff))$$
Returns

the int value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.291.3.9 `virtual std::string decaf::io::DataInputStream::readLine () throw (decaf::io::IOException) [virtual]`

Reads the next line of text from the input stream.

It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character ' is

encountered, it is discarded and reading ceases. If the character " is encountered, it is discarded and, if the following byte converts to the character ' is

encountered, then that is discarded also; reading then ceases. If end of file is encountered before either of the characters ' and " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

Returns

the next line of text read from the input stream or empty string if at EOF.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

6.291.3.10 `virtual long long decaf::io::DataInputStream::readLong () throw (decaf::io::IOException, decaf::io::EOFException) [virtual]`

Reads eight input bytes and returns a long value.

Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth byte. The value returned is:

$$(((\text{long})(a \ \& \ 0\text{xff}) \ll 56) \mid ((\text{long})(b \ \& \ 0\text{xff}) \ll 48) \mid ((\text{long})(c \ \& \ 0\text{xff}) \ll 40) \mid ((\text{long})(d \ \& \ 0\text{xff}) \ll 32) \mid ((\text{long})(e \ \& \ 0\text{xff}) \ll 24) \mid ((\text{long})(f \ \& \ 0\text{xff}) \ll 16) \mid ((\text{long})(g \ \& \ 0\text{xff}) \ll 8) \mid ((\text{long})(h \ \& \ 0\text{xff})))$$
Returns

the 64 bit long long read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.291.3.11 `virtual short decaf::io::DataInputStream::readShort () throw (decaf::io::IOException, decaf::io::EOFException) [virtual]`

Reads two input bytes and returns a short value.

Let a be the first byte read and b be the second byte. The value returned is:

$$(\text{short})((a \ll 8) \mid (b \ \& \ 0\text{xff}))$$
Returns

the 16 bit short value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.291.3.12 `virtual std::string decaf::io::DataInputStream::readString () throw (decaf::io::IOException, decaf::io::EOFException) [virtual]`

Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

Returns

string object containing the string read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.291.3.13 virtual unsigned char decaf::io::DataInputStream::readUnsignedByte () throw (decaf::io::IOException, decaf::io::EOFException) [virtual]

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

Returns

the 8 bit unsigned value read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.291.3.14 virtual unsigned short decaf::io::DataInputStream::readUnsignedShort () throw (decaf::io::IOException, decaf::io::EOFException) [virtual]

Reads two input bytes and returns an int value in the range 0 through 65535.

Let a be the first byte read and b be the second byte. The value returned is:

$((a \& 0xff) \ll 8) | (b \& 0xff)$

Returns

the 16 bit unsigned short read.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

6.291.3.15 virtual std::string decaf::io::DataInputStream::readUTF () throw (decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException) [virtual]

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.

This method reads String value written from a Java **DataOutputStream** (p. 1473) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

Returns

The decoded string read from stream.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

EOFException (p. 1707) if the end of input is reached.

UTFDataFormatException (p. 3703) if the bytes are not valid modified UTF-8 values.

6.291.3.16 virtual long long decaf::io::DataInputStream::skipBytes (long long num) throw (io::IOException) [virtual]

Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. This method never throws an ***EOFException*** (p.1707). The actual number of bytes skipped is returned.

Parameters

num The number of bytes to skip over.

Returns

the total number of bytes skipped.

Exceptions

IOException (p. 2003) if an I/O Error occurs.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataInputStream.h**

6.292 decaf::io::DataOutput Class Reference

The **DataOutput** (p. 1468) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.

```
#include <src/main/decaf/io/DataOutput.h>
```

Public Member Functions

- virtual **~DataOutput** ()
- virtual void **writeBoolean** (bool value)=0 throw (decaf::io::IOException)
Writes a boolean to the underlying output stream as a 1-byte value.
- virtual void **writeByte** (unsigned char value)=0 throw (decaf::io::IOException)
Writes out a byte to the underlying output stream as a 1-byte value.
- virtual void **writeShort** (short value)=0 throw (decaf::io::IOException)
Writes a short to the underlying output stream as two bytes, high byte first.

- virtual void **writeUnsignedShort** (unsigned short value)=0 throw (decaf::io::IOException)
Writes a unsigned short to the bytes message stream as a 2 byte value.
- virtual void **writeChar** (char value)=0 throw (decaf::io::IOException)
Writes out a char to the underlying output stream as a one byte value If no exception is thrown, the counter written is incremented by 1.
- virtual void **writeInt** (int value)=0 throw (decaf::io::IOException)
Writes an int to the underlying output stream as four bytes, high byte first.
- virtual void **writeLong** (long long value)=0 throw (decaf::io::IOException)
Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.
- virtual void **writeFloat** (float value)=0 throw (decaf::io::IOException)
Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.
- virtual void **writeDouble** (double value)=0 throw (decaf::io::IOException)
Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.
- virtual void **writeBytes** (const std::string &value)=0 throw (decaf::io::IOException)
Writes out the string to the underlying output stream as a sequence of bytes.
- virtual void **writeChars** (const std::string &value)=0 throw (decaf::io::IOException)
Writes a string to the underlying output stream as a sequence of characters.
- virtual void **writeUTF** (const std::string &value)=0 throw (decaf::io::IOException, decaf::io::UTFDataFormatException)
Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes.

6.292.1 Detailed Description

The **DataOutput** (p. 1468) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream. There is also a facility for converting Strings into the Java standard modified UTF-8 format and writing the resulting series of bytes.

If a method in this interface encounters an error while writing it will throw an **IOException** (p. 2003).

See also

DataInput (p. 1452)

DataOutputStream (p. 1473)

Since

1.0

6.292.2 Constructor & Destructor Documentation

6.292.2.1 `virtual decaf::io::DataOutput::~~DataOutput () [inline, virtual]`

6.292.3 Member Function Documentation

6.292.3.1 `virtual void decaf::io::DataOutput::writeBoolean (bool value) throw (decaf::io::IOException) [pure virtual]`

Writes a boolean to the underlying output stream as a 1-byte value.

The value true is written out as the value (byte)1; the value false is written out as the value (byte)0. If no exception is thrown, the counter written is incremented by 1.

Parameters

value The boolean to write as a byte (1=true, 0=false).

Exceptions

IOException (p. 2003) if an I/O error is encountered.

6.292.3.2 `virtual void decaf::io::DataOutput::writeByte (unsigned char value) throw (decaf::io::IOException) [pure virtual]`

Writes out a byte to the underlying output stream as a 1-byte value.

If no exception is thrown, the counter written is incremented by 1.

Parameters

value The unsigned char value to write.

Exceptions

IOException (p. 2003) if an I/O error is encountered.

6.292.3.3 `virtual void decaf::io::DataOutput::writeBytes (const std::string & value) throw (decaf::io::IOException) [pure virtual]`

Writes out the string to the underlying output stream as a sequence of bytes.

Each character in the string is written out, in sequence, by discarding its high eight bits. If no exception is thrown, the counter written is incremented by the length of value. The value written does not include a trailing null as that is not part of the sequence of bytes, if the null is needed, then use the writeChars method.

Parameters

value The vector of bytes to write.

Exceptions

IOException (p. 2003) if an I/O error is encountered.

6.292.3.4 virtual void decaf::io::DataOutput::writeChar (char *value*) throw (decaf::io::IOException) [pure virtual]

Writes out a char to the underlying output stream as a one byte value. If no exception is thrown, the counter written is incremented by 1.

Parameters

value The signed char value to write.

Exceptions

IOException (p. 2003) if an I/O error is encountered.

6.292.3.5 virtual void decaf::io::DataOutput::writeChars (const std::string & *value*) throw (decaf::io::IOException) [pure virtual]

Writes a string to the underlying output stream as a sequence of characters.

Each character is written to the data output stream as if by the writeChar method. If no exception is thrown, the counter written is incremented by the length of value. The trailing NULL character is written by this method.

Parameters

value The string value to write as raw bytes.

Exceptions

IOException (p. 2003) if an I/O error is encountered.

6.292.3.6 virtual void decaf::io::DataOutput::writeDouble (double *value*) throw (decaf::io::IOException) [pure virtual]

Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.

If no exception is thrown, the counter written is incremented by 8.

Parameters

value The 64bit double value to write.

Exceptions

IOException (p. 2003) if an I/O error is encountered.

6.292.3.7 virtual void decaf::io::DataOutput::writeFloat (float *value*) throw (decaf::io::IOException) [pure virtual]

Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.

If no exception is thrown, the counter written is incremented by 4.

Parameters

value The 32bit floating point value to write.

Exceptions

IOException (p. 2003) if an I/O error is encountered.

6.292.3.8 `virtual void decaf::io::DataOutput::writeInt (int value) throw (decaf::io::IOException) [pure virtual]`

Writes an int to the underlying output stream as four bytes, high byte first.

If no exception is thrown, the counter written is incremented by 4.

Parameters

value The signed integer value to write.

Exceptions

IOException (p. 2003) if an I/O error is encountered.

6.292.3.9 `virtual void decaf::io::DataOutput::writeLong (long long value) throw (decaf::io::IOException) [pure virtual]`

Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.

If no exception is thrown, the counter written is incremented by 8.

Parameters

value The signed 64bit long value to write.

Exceptions

IOException (p. 2003) if an I/O error is encountered.

6.292.3.10 `virtual void decaf::io::DataOutput::writeShort (short value) throw (decaf::io::IOException) [pure virtual]`

Writes a short to the underlying output stream as two bytes, high byte first.

If no exception is thrown, the counter written is incremented by 2.

Parameters

value The signed short value to write.

Exceptions

IOException (p. 2003) if an I/O error is encountered.

6.292.3.11 `virtual void decaf::io::DataOutput::writeUnsignedShort (unsigned short value) throw (decaf::io::IOException)` [pure virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters

value The unsigned short to write to the stream.

Exceptions

IOException (p. 2003) if an I/O error is encountered.

6.292.3.12 `virtual void decaf::io::DataOutput::writeUTF (const std::string & value) throw (decaf::io::IOException, decaf::io::UTFDataFormatException)` [pure virtual]

Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes.

The first two bytes written are indicate its encoded length followed by the rest of the string's characters encoded as modified UTF-8. The length represent the encoded length of the data not the actual length of the string.

Parameters

value The string value value to write as modified UTF-8.

Exceptions

IOException (p. 2003) if an I/O error is encountered.

UTFDataFormatException (p. 3703) if the encoded size if greater than 65535

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataOutput.h`

6.293 decaf::io::DataOutputStream Class Reference

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

```
#include <src/main/decaf/io/DataOutputStream.h>
```

Inheritance diagram for decaf::io::DataOutputStream:

Public Member Functions

- **DataOutputStream** (`OutputStream *outputStream`, `bool own=false`)
Creates a new data output stream to write data to the specified underlying output stream.

- virtual `~DataOutputStream ()`
- virtual long long **size** () const
Returns the current value of the counter written, the number of bytes written to this data output stream so far.
- virtual void **writeBoolean** (bool value) throw (IOException)
- virtual void **writeByte** (unsigned char value) throw (IOException)
- virtual void **writeShort** (short value) throw (IOException)
- virtual void **writeUnsignedShort** (unsigned short value) throw (IOException)
- virtual void **writeChar** (char value) throw (IOException)
- virtual void **writeInt** (int value) throw (IOException)
- virtual void **writeLong** (long long value) throw (IOException)
- virtual void **writeFloat** (float value) throw (IOException)
- virtual void **writeDouble** (double value) throw (IOException)
- virtual void **writeBytes** (const std::string &value) throw (IOException)
- virtual void **writeChars** (const std::string &value) throw (IOException)
- virtual void **writeUTF** (const std::string &value) throw (IOException, UTFDataFormatException)

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw (decaf::io::IOException)
- virtual void **doWriteArrayBounded** (const unsigned char ***buffer**, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Protected Attributes

- long long **written**
- unsigned char **buffer** [8]

6.293.1 Detailed Description

A data output stream lets an application write primitive Java data types to an output stream in a portable way. An application can then use a data input stream to read the data back in.

6.293.2 Constructor & Destructor Documentation

6.293.2.1 `decaf::io::DataOutputStream::DataOutputStream (OutputStream * outputStream, bool own = false)`

Creates a new data output stream to write data to the specified underlying output stream.

Parameters

outputStream a stream to wrap with this one.

own true if this objects owns the stream that it wraps.

6.293.2.2 `virtual decaf::io::DataOutputStream::~~DataOutputStream ()`
[virtual]

6.293.3 Member Function Documentation

6.293.3.1 `virtual void decaf::io::DataOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1779).

6.293.3.2 `virtual void decaf::io::DataOutputStream::doWriteByte (unsigned char value) throw (decaf::io::IOException)` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1779).

6.293.3.3 `virtual long long decaf::io::DataOutputStream::size () const` [inline, virtual]

Returns the current value of the counter written, the number of bytes written to this data output stream so far.

If the counter overflows, it will be wrapped to `decaf::lang::Long::MAX_VALUE` (p. 2281).

Returns

the value of the written field.

- 6.293.3.4 `virtual void decaf::io::DataOutputStream::writeBoolean (bool value) throw (IOException) [virtual]`
- 6.293.3.5 `virtual void decaf::io::DataOutputStream::writeByte (unsigned char value) throw (IOException) [virtual]`
- 6.293.3.6 `virtual void decaf::io::DataOutputStream::writeBytes (const std::string & value) throw (IOException) [virtual]`
- 6.293.3.7 `virtual void decaf::io::DataOutputStream::writeChar (char value) throw (IOException) [virtual]`
- 6.293.3.8 `virtual void decaf::io::DataOutputStream::writeChars (const std::string & value) throw (IOException) [virtual]`
- 6.293.3.9 `virtual void decaf::io::DataOutputStream::writeDouble (double value) throw (IOException) [virtual]`
- 6.293.3.10 `virtual void decaf::io::DataOutputStream::writeFloat (float value) throw (IOException) [virtual]`
- 6.293.3.11 `virtual void decaf::io::DataOutputStream::writeInt (int value) throw (IOException) [virtual]`
- 6.293.3.12 `virtual void decaf::io::DataOutputStream::writeLong (long long value) throw (IOException) [virtual]`
- 6.293.3.13 `virtual void decaf::io::DataOutputStream::writeShort (short value) throw (IOException) [virtual]`
- 6.293.3.14 `virtual void decaf::io::DataOutputStream::writeUnsignedShort (unsigned short value) throw (IOException) [virtual]`
- 6.293.3.15 `virtual void decaf::io::DataOutputStream::writeUTF (const std::string & value) throw (IOException, UTFDataFormatException) [virtual]`

6.293.4 Field Documentation

- 6.293.4.1 `unsigned char decaf::io::DataOutputStream::buffer[8] [protected]`
- 6.293.4.2 `long long decaf::io::DataOutputStream::written [protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataOutputStream.h`

6.294 activemq::commands::DataResponse Class Reference

```
#include <src/main/activemq/commands/DataResponse.h>
```

Inheritance diagram for activemq::commands::DataResponse:

Public Member Functions

- **DataResponse** ()
- virtual **~DataResponse** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **DataResponse** * **cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **DataStructure** > & **getData** () const

- virtual **Pointer**< **DataStructure** > & **getData** ()

- virtual void **setData** (const **Pointer**< **DataStructure** > &data)

Static Public Attributes

- static const unsigned char **ID_DATARESPONSE** = 32

Protected Attributes

- **Pointer**< **DataStructure** > data

6.294.1 Constructor & Destructor Documentation

6.294.1.1 `activemq::commands::DataResponse::DataResponse ()`

6.294.1.2 `virtual activemq::commands::DataResponse::~~DataResponse ()`
[virtual]

6.294.2 Member Function Documentation

6.294.2.1 `virtual DataResponse* activemq::commands::DataResponse::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p.3077).

6.294.2.2 `virtual void activemq::commands::DataResponse::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::Response` (p.3078).

6.294.2.3 `virtual bool activemq::commands::DataResponse::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p.1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Response` (p.3078).

- 6.294.2.4** `virtual Pointer<DataStructure>& activemq::commands::DataResponse::getData () [virtual]`
- 6.294.2.5** `virtual const Pointer<DataStructure>& activemq::commands::DataResponse::getData () const [virtual]`
- 6.294.2.6** `virtual unsigned char activemq::commands::DataResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaller share.

Returns

new **DataStructure** (p. 1553) type copy.

Reimplemented from **activemq::commands::Response** (p. 3078).

- 6.294.2.7** `virtual void activemq::commands::DataResponse::setData (const Pointer< DataStructure > & data) [virtual]`
- 6.294.2.8** `virtual std::string activemq::commands::DataResponse::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3079).

6.294.3 Field Documentation

- 6.294.3.1** `Pointer<DataStructure> activemq::commands::DataResponse::data [protected]`
- 6.294.3.2** `const unsigned char activemq::commands::DataResponse::ID_ - DATARESPONSE = 32 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataResponse.h`

6.295 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1479).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller`:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.295.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p.1479). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.295.2 Constructor & Destructor Documentation

6.295.2.1 `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::DataResponseMar
() [inline]`

6.295.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::~~DataResponseM
() [inline, virtual]`

6.295.3 Member Function Documentation

6.295.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller`
(p. 3095).

6.295.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller`
(p. 3095).

6.295.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3095).

6.295.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3096).

6.295.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3096).

6.295.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3097).

6.295.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3097).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h`

6.296 `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `DataResponseMarshaller` (p. 1483).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller`:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.296.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p.1483). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.296.2 Constructor & Destructor Documentation

6.296.2.1 `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::DataResponseMar
() [inline]`

6.296.2.2 `virtual
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::~~DataResponseM
() [inline, virtual]`

6.296.3 Member Function Documentation

6.296.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller`
(p. 3108).

6.296.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller`
(p. 3109).

6.296.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109).

6.296.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109).

6.296.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3110).

6.296.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3110).

6.296.3.7 `virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3111).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h`

6.297 `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `DataResponseMarshaller` (p. 1487).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller`:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.297.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p.1487). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.297.2 Constructor & Destructor Documentation

6.297.2.1 `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::DataResponseMar
() [inline]`

6.297.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::~~DataResponseM
() [inline, virtual]`

6.297.3 Member Function Documentation

6.297.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3090).

6.297.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3091).

6.297.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3091).

6.297.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3091).

6.297.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3092).

6.297.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3092).

6.297.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3093).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h`

6.298 `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `DataResponseMarshaller` (p. 1491).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller`:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.298.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1491). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.298.2 Constructor & Destructor Documentation

6.298.2.1 `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::DataResponseMar
() [inline]`

6.298.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::~~DataResponseM
() [inline, virtual]`

6.298.3 Member Function Documentation

6.298.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller`
(p. 3099).

6.298.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller`
(p. 3100).

6.298.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3100).

6.298.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3100).

6.298.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3101).

6.298.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3101).

6.298.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3102).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h`

6.299 **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1495).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller`:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.299.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p.1495). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.299.2 Constructor & Destructor Documentation

6.299.2.1 `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::DataResponseMar
() [inline]`

6.299.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::~~DataResponseM
() [inline, virtual]`

6.299.3 Member Function Documentation

6.299.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller`
(p. 3086).

6.299.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller`
(p. 3086).

6.299.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3086).

6.299.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3087).

6.299.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3087).

6.299.3.6 virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3088).

6.299.3.7 virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3088).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DataResponseMarshaller.h**

6.300 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1499).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller`:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.300.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p.1499). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.300.2 Constructor & Destructor Documentation

6.300.2.1 `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::DataResponseMar
() [inline]`

6.300.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::~~DataResponseM
() [inline, virtual]`

6.300.3 Member Function Documentation

6.300.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3104).

6.300.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3104).

6.300.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3104).

6.300.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3105).

6.300.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3105).

6.300.3.6 virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3106).

6.300.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3106).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DataResponseMarshaller.h**

6.301 activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference

Base class for all classes that marshal commands for Openwire.

```
#include <src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::DataStreamMarshaller`:

Public Member Functions

- virtual `~DataStreamMarshaller ()`
- virtual unsigned char `getDataStructureType ()` const =0
Gets the DataStructureType that this class marshals/unmarshals.
- virtual `commands::DataStructure * createObject ()` const =0
Creates a new instance of the class that this class is a marshaling director for.
- virtual int `tightMarshal (OpenWireFormat *format, commands::DataStructure *command, utils::BooleanStream *bs)=0` throw (`decaf::io::IOException`)
Tight Marshal to the given stream.
- virtual void `tightMarshal2 (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataOutputStream *ds, utils::BooleanStream *bs)=0` throw (`decaf::io::IOException`)
Tight Marshal to the given stream.
- virtual void `tightUnmarshal (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataInputStream *dis, utils::BooleanStream *bs)=0` throw (`decaf::io::IOException`)
Tight Un-marshal to the given stream.
- virtual void `looseMarshal (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataOutputStream *ds)=0` throw (`decaf::io::IOException`)
Tight Marshal to the given stream.
- virtual void `looseUnmarshal (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataInputStream *dis)=0` throw (`decaf::io::IOException`)
Loose Un-marshal to the given stream.

6.301.1 Detailed Description

Base class for all classes that marshal commands for Openwire.

6.301.2 Constructor & Destructor Documentation

6.301.2.1 virtual
activemq::wireformat::openwire::marshal::DataStreamMarshaller::~~DataStreamMarshaller
() [inline, virtual]

6.301.3 Member Function Documentation

6.301.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::DataStreamMarshaller::createObject
() const [pure virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (p. 176), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (p. 215), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (p. 334), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (p. 360), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (p. 404), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (p. 446), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 505), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 558), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 590), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 618), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 646), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 810), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 841), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1186), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1217), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1247), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1277), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1320), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1347), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1379), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1406), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1439), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1501), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1631), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1664), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1744), activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1832), activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1976), activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 2038), activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 2067), activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2089), activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2120), activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2146), activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (p. 2179), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (p. 2225), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller

(p. 2416), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2455), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2484), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2520), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2585), `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2638), `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2753), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2864), `activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 2895), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2911), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3004), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3020), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3051), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3104), `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3186), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3201), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3261), `activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3443), `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3604), `activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller` (p. 3745), `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3782), `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 346), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 372), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 416), `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 458), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 517), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 570), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 598), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 630), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 658), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 822), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 853), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1198), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1205), `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1235), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1265), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1308), `activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1335), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1367), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1394), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1427), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1489), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1619), `activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1652), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1728), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1820), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1964), `activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2022), `activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2051), `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2073), `activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2104), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2130), `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`

(p. 2167), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 2209), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2404), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2439), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (p. 2472), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2500), activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2569), activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2618), activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2736), activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2844), activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2875), activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2907), activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2992), activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (p. 3028), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (p. 3055), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3090), activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3166), activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3209), activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3257), activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3459), activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3620), activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3737), activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3774), activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 172), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (p. 211), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (p. 330), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (p. 356), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (p. 400), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller (p. 442), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (p. 501), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller (p. 554), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller (p. 582), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (p. 610), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller (p. 638), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 802), activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 833), activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1178), activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1209), activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1239), activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1269), activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1312), activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1339), activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1371), activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1398), activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1431), activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1493), activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1623), activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1656), activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1732), activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1824), activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1968), activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2030), activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2055), activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2077), activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2108),

activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2134),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 2163), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 2213), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2408), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2443), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2476), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2512),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2577), ac-
 tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2630),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2745),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2852),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2883),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2919),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3000), ac-
 tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 3024), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 3059), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3099),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3182), ac-
 tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3205), ac-
 tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3269), ac-
 tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3439),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3608),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3749),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3786),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 180), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 219), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 338), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 364), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 408), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 450), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 509), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 562), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
 (p. 586), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
 (p. 614), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
 (p. 642), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 806),
 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 837), ac-
 tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1182),
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1213),
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1243),
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1273),
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1316),
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1343),
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1375), ac-
 tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1402), ac-
 tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1435),
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1497),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1627),
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1660), ac-
 tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1740),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1828),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1972),
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2034),
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2063),

activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2085), activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2116), activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2138), activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (p. 2175), activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (p. 2221), activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2412), activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2451), activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (p. 2480), activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2504), activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2581), activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2634), activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2749), activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2848), activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 2879), activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2903), activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3012), activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (p. 3040), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (p. 3047), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3086), activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3170), activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3213), activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3273), activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3451), activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3616), activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3741), activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3778), activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 188), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (p. 223), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (p. 342), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (p. 368), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (p. 412), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (p. 454), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller (p. 513), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller (p. 566), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller (p. 594), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller (p. 622), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller (p. 650), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 814), activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 845), activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1190), activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1221), activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1251), activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1281), activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1324), activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1351), activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1383), activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1410), activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1443), activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1481), activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1639), activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1668), activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1736), activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1836), activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1980),

activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2026),
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2047),
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2093),
 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2112),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2142),
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
 (p. 2171), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller
 (p. 2217), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
 (p. 2420), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
 (p. 2447), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2488), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2508),
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2573),
 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2626),
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2741),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2856),
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 2887),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2915),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3008),
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
 (p. 3036), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
 (p. 3067), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3095),
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3178),
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3197),
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3265),
 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3447),
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3600),
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3729),
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3790),
 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller
 (p. 192), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller
 (p. 227), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller
 (p. 350), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller
 (p. 376), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller
 (p. 420), activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller
 (p. 462), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller
 (p. 521), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller
 (p. 574), activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller
 (p. 602), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller
 (p. 626), activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller
 (p. 654), activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 818),
 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 849),
 activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1194),
 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1225),
 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1255),
 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1285),
 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1328),
 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1355),
 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1387),
 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1414),
 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1447),
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1485),
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1635),
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1648),
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1724),

activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1816),
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 1960),
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2018),
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2059),
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2081),
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2100),
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2126),
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller
 (p. 2159), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller
 (p. 2205), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller
 (p. 2400), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller
 (p. 2459), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller
 (p. 2468), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2516),
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2589),
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2622),
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 2732),
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 2860),
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 2891),
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 2923),
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 2996),
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller
 (p. 3032), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller
 (p. 3063), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3108),
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (p. 3174),
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3193),
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3253),
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3455),
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3612),
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 3733), and
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 3770).

6.301.3.2 virtual unsigned char ac-

tivemq::wireformat::openwire::marshal::DataStreamMarshaller::getDataStructureType
 () const [pure virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller
 (p. 176), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller
 (p. 215), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller
 (p. 334), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller
 (p. 360), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller
 (p. 404), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller
 (p. 446), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller
 (p. 505), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller
 (p. 558), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller
 (p. 590), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller
 (p. 618), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller
 (p. 646), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 810),

activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 841),
 activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1186),
 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1217),
 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1247),
 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1277),
 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1320),
 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1347),
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1379),
 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1406),
 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1439),
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1501),
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1631),
 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1664),
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1744),
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1832),
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1976),
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 2038),
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 2067),
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2089),
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2120),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2146),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (p. 2179),
 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (p. 2225),
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2416),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2455),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 2484),
 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2520),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2585),
 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2638),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2753),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2864),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2895),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2911),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3004),
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 3021),
 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 3051),
 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3104),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3186),
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3201),
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3261),
 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3444),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3604),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3745),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3782),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 184),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 231),
 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 346),
 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 372),
 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 416),
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 458),
 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 517),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 570),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 598),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller

(p. 630), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller`
 (p. 658), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 822),
`activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 853), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1198),
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1206),
`activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1235),
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1265),
`activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1308),
`activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1335),
`activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1367), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1395), `ac-`
`tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1427),
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1489),
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1620),
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1652), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1728),
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1820),
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1964),
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2022),
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2051),
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2073), `ac-`
`tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2104),
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2130),
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`
 (p. 2167), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller`
 (p. 2209), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller`
 (p. 2404), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`
 (p. 2439), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`
 (p. 2472), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2500),
`activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2570), `ac-`
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2618),
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2737),
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2844),
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 2875),
`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 2907),
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2993), `ac-`
`tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`
 (p. 3028), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`
 (p. 3055), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3091),
`activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 3166), `ac-`
`tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3209), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3257), `ac-`
`tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3459),
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3620),
`activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller` (p. 3737),
`activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3774),
`activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`
 (p. 172), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`
 (p. 211), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`
 (p. 330), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`
 (p. 356), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`
 (p. 400), `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller`
 (p. 442), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`
 (p. 501), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller`

(p. 554), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller`
 (p. 582), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`
 (p. 611), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`
 (p. 638), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 802),
`activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 833), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1178),
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1209),
`activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1239),
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1269),
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1312),
`activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1339),
`activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1371), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1398), `ac-`
`tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1431),
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1493),
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1623),
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1656), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1732),
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1824),
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1968),
`activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2030),
`activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2055),
`activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2077), `ac-`
`tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2108),
`activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2134),
`activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller`
 (p. 2164), `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller`
 (p. 2213), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller`
 (p. 2408), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller`
 (p. 2443), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`
 (p. 2476), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2512),
`activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2577), `ac-`
`tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2630),
`activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2745),
`activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2852),
`activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 2883),
`activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 2919),
`activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3000), `ac-`
`tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller`
 (p. 3024), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`
 (p. 3059), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3100),
`activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3182), `ac-`
`tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3205), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3269), `ac-`
`tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3440),
`activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3608),
`activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 3749),
`activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3786),
`activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`
 (p. 180), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`
 (p. 219), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`
 (p. 338), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`
 (p. 364), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`
 (p. 408), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`

(p. 450), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`
 (p. 509), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`
 (p. 562), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`
 (p. 586), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`
 (p. 614), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`
 (p. 642), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 806),
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 837), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1182),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1213),
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1243),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1273),
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1316),
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1343),
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1375), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1402), `ac-`
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1435),
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1497),
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1627),
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1660), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1740),
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1828),
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1972),
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2034),
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2063),
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2085), `ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2116),
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2138),
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`
 (p. 2175), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`
 (p. 2221), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`
 (p. 2412), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`
 (p. 2451), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`
 (p. 2480), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2504),
`activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2581), `ac-`
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2634),
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2749),
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2848),
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 2879),
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2903),
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3012), `ac-`
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`
 (p. 3040), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`
 (p. 3047), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3086),
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 3170), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3213), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3273), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3451),
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3616),
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3741),
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3778),
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`
 (p. 188), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`
 (p. 223), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`
 (p. 342), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`

(p. 368), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`
 (p. 412), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`
 (p. 454), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`
 (p. 513), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`
 (p. 566), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`
 (p. 594), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`
 (p. 622), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`
 (p. 650), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 814),
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 845), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1190),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1221),
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1251),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1281),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1324),
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1351),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1383), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1410), `ac-`
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1443),
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1481),
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1639),
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1668), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1736),
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1836),
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1980),
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2026),
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2047),
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2093), `ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2112),
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2142),
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`
 (p. 2171), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`
 (p. 2217), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`
 (p. 2420), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`
 (p. 2447), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`
 (p. 2488), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2508),
`activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2573), `ac-`
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2626),
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2741),
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2856),
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 2887),
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2915), `ac-`
`tivemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3008), `ac-`
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`
 (p. 3036), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`
 (p. 3067), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3095),
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3178), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3197), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3265), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3447),
`activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3600),
`activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3729),
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3790),
`activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller`
 (p. 192), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller`

(p. 227), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`
 (p. 350), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`
 (p. 376), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`
 (p. 420), `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller`
 (p. 462), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller`
 (p. 521), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller`
 (p. 574), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller`
 (p. 602), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller`
 (p. 626), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`
 (p. 654), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 818),
`activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 849), `ac-`
`tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1194),
`activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1225),
`activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1255),
`activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1285),
`activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1328),
`activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1355),
`activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1387), `ac-`
`tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1414), `ac-`
`tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1447),
`activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1485),
`activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1635),
`activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1648), `ac-`
`tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1724),
`activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1816),
`activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1960),
`activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2019),
`activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2059),
`activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2081), `ac-`
`tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2100),
`activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2127),
`activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller`
 (p. 2160), `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller`
 (p. 2205), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller`
 (p. 2401), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller`
 (p. 2459), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller`
 (p. 2468), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2516),
`activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2589), `ac-`
`tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2622),
`activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2732),
`activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 2860),
`activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 2891),
`activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 2923),
`activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 2996), `ac-`
`tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller`
 (p. 3032), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller`
 (p. 3063), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109),
`activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3174), `ac-`
`tivemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3193), `ac-`
`tivemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3254), `ac-`
`tivemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3455),
`activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3612), `ac-`
`tivemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 3733), and
`activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 3770).

6.301.3.3 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) throw (decaf::io::IOException)
[pure virtual]

Tight Marshal to the given stream.

Parameters

format - The OpenWireFormat properties

command - the object to Marshal

ds - DataOutputStream to marshal to

Exceptions

IOException if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (p. 176), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (p. 215), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (p. 295), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (p. 334), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (p. 361), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (p. 405), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (p. 447), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 505), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (p. 531), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 558), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 590), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 619), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 646), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (p. 716), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 810), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 841), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1186), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1218), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1247), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1277), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1320), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1347), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1379), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1407), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1439), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1501), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1632), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1664), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1744), activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1832), activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1976), activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 2038), activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 2067), activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2089), activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2120), activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2147),

activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (p. 2180),
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (p. 2226),
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2417),
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2456),
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 2484),
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2520),
activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2541),
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2586),
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2638),
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2754),
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2864),
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2895),
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2912),
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3005),
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 3021),
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 3052),
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3104),
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3186),
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3201),
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3262),
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3444),
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3577),
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3604),
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3745),
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3783),
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 184),
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 231),
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller (p. 307),
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 346),
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 373),
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 417),
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 459),
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 517),
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller (p. 542),
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 570),
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 598),
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 631),
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 658),
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller (p. 736),
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 822),
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 853),
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1198),
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1206),
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1236),
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1265),
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1308),
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1336),
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1367),
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1395),
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1427),
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1489),
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1620),
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1652),
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1728),

activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1820),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1964),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2023),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2051),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2074),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2104),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2131),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
 (p. 2168), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller
 (p. 2210), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
 (p. 2405), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
 (p. 2440), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2472), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2501),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2533),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2570),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2618),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2737),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2844),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2876),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2908),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2993),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 3029), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 3056), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3091),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3167),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3209),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3258),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3459),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3581),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3620),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3737),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3775),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 172), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 211), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
 (p. 291), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 330), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 357), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 401), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 443), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 501), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller
 (p. 528), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 554), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 582), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 611), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 638), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
 (p. 702), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 803),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 833),
 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1178),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1210),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1239),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1269),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1312),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1340),

activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1371),
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1399),
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1431),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1493),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1624),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1656),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1732),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1824),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1968),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2030),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2055),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2077),
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2108),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2135),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 2164), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 2214), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2409), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2444), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2476), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2512),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2529),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2578),
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2630),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2745),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2852),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2883),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2920),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3001),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 3025), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 3060), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3100),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3182),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3205),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3270),
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3440),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3585),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3608),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3749),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3787),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 180), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 219), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
 (p. 299), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 338), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 365), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 409), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 451), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 509), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
 (p. 535), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 562), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
 (p. 586), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
 (p. 615), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
 (p. 642), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller
 (p. 709), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 806),

activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 837),
 activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1182),
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1214),
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1243),
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1273),
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1316),
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1343),
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1375),
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1403),
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1435),
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1497),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1628),
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1660),
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1740),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1828),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1972),
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2034),
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2063),
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2085),
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2116),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2139),
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (p. 2176),
 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (p. 2222),
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2413),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2452),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (p. 2480),
 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2505),
 activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2537),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2582),
 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2634),
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2750),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2848),
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 2879),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2904),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3013),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (p. 3041),
 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (p. 3048),
 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3086),
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3171),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3213),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3274),
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3451),
 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3588),
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3616),
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3741),
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3779),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 188),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (p. 223),
 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller (p. 303),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (p. 342),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (p. 369),
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (p. 413),
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (p. 455),
 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

(p. 513), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`
(p. 539), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`
(p. 566), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`
(p. 594), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`
(p. 623), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`
(p. 650), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`
(p. 722), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 814),
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 845), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1190),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1222),
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1251),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1281),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1324),
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1351),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1383), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1411), `ac-`
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1443),
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1481),
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1640),
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1668), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1736),
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1836),
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1980),
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2026),
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2047),
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2093), `ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2112),
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2143),
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`
(p. 2172), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`
(p. 2218), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`
(p. 2421), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`
(p. 2448), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`
(p. 2488), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2508),
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2524), `ac-`
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2574), `ac-`
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2626),
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2741),
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2856),
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 2887),
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2916),
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3009), `ac-`
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`
(p. 3037), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`
(p. 3068), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3095),
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3178), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3197), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3266), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3448),
`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3574),
`activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3600),
`activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3729),
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3791),
`activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller`

(p. 192), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller`
 (p. 227), `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller`
 (p. 311), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`
 (p. 350), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`
 (p. 377), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`
 (p. 421), `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller`
 (p. 463), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller`
 (p. 521), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller`
 (p. 546), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller`
 (p. 574), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller`
 (p. 602), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller`
 (p. 627), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`
 (p. 654), `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller`
 (p. 729), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 818),
`activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 849), `ac-`
`tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1194),
`activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1226),
`activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1255),
`activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1285),
`activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1328),
`activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1355),
`activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1387), `ac-`
`tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1415), `ac-`
`tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1447),
`activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1485),
`activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1636),
`activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1649), `ac-`
`tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1724),
`activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1816),
`activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1960),
`activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2019),
`activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2059),
`activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2081), `ac-`
`tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2100),
`activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2127),
`activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller`
 (p. 2160), `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller`
 (p. 2206), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller`
 (p. 2401), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller`
 (p. 2460), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller`
 (p. 2468), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2516),
`activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2545), `ac-`
`tivemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2590), `ac-`
`tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2622),
`activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2733),
`activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 2860),
`activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 2891),
`activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 2924),
`activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 2997), `ac-`
`tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller`
 (p. 3033), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller`
 (p. 3064), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109),
`activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3174), `ac-`
`tivemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3193), `ac-`

activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3254), activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3455), activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller (p. 3592), activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3612), activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 3733), and activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 3771).

6.301.3.4 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) throw (decaf::io::IOException)
[pure virtual]

Loose Un-marhsal to the given stream.

Parameters

format - The OpenWireFormat properties
command - the object to Un-Marshal
dis - the DataInputStream to Un-Marshal from

Exceptions

IOException if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (p. 177), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (p. 216), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (p. 295), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (p. 335), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (p. 361), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (p. 405), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (p. 447), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 506), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (p. 532), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 559), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 591), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 619), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 647), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (p. 717), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 811), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 842), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1187), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1218), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1248), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1278), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1321), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1348), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1380), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1407), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1440), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1502), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1632), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1665), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1745),

activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1833),
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1977),
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 2039),
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 2068),
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2090),
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2121),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2147),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 2180), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 2226), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2417), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2456), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2485), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2521),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2541),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2586),
 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2639),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2754),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2865),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2896),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2912),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3005),
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 3021), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 3052), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3105),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3187),
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3202),
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3262),
 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3444),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3578),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3605),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3746),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3783),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 185), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 232), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 307), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 347), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 373), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 417), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 459), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 518), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 543), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 571), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 599), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 631), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 659), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 737), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 823),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 854),
 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1199),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1206),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1236),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1266),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1309),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1336),

activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1368),
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1395),
 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1428),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1490),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1620),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1653),
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1729),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1821),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1965),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2023),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2052),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2074),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2105),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2131),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
 (p. 2168), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller
 (p. 2210), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
 (p. 2405), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
 (p. 2440), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2473), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2501),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2533),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2570),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2619),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2737),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2845),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2876),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2908),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2993),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 3029), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 3056), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3091),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3167),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3210),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3258),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3460),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3581),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3621),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3738),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3775),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 173), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 212), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
 (p. 291), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 331), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 357), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 401), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 443), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 502), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller
 (p. 528), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 555), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 583), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 611), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 639), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
 (p. 703), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 803),

activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 834),
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1179),
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1210),
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1240),
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1270),
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1313),
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1340),
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1372),
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1399),
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1432),
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1494),
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1624),
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1657),
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1733),
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1825),
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1969),
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2031),
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2056),
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2078),
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2109),
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2135),
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (p. 2164),
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (p. 2214),
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2409),
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2444),
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (p. 2477),
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2513),
activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2529),
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2578),
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2631),
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2746),
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2853),
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2884),
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2920),
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3001),
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (p. 3025),
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (p. 3060),
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3100),
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3183),
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3206),
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3270),
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3440),
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3585),
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3609),
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3750),
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3787),
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller (p. 181),
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller (p. 220),
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller (p. 299),
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller (p. 339),
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller (p. 365),
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller (p. 409),
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller (p. 451),
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller

(p. 510), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller`
 (p. 536), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`
 (p. 563), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`
 (p. 587), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`
 (p. 615), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`
 (p. 643), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`
 (p. 710), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 807),
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 838), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1183),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1214),
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1244),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1274),
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1317),
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1344),
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1376), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1403), `ac-`
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1436),
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1498),
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1628),
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1661), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1741),
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1829),
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1973),
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2035),
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2064),
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2086), `ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2117),
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2139),
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`
 (p. 2176), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`
 (p. 2222), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`
 (p. 2413), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`
 (p. 2452), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`
 (p. 2481), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2505),
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2537), `ac-`
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2582), `ac-`
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2635),
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2750),
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2849),
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 2880),
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2904),
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3013), `ac-`
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`
 (p. 3041), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`
 (p. 3048), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3087),
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 3171), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3214), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3274), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3452),
`activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3589),
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3617),
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3742),
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3779),
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`

(p. 189), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`
 (p. 224), `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller`
 (p. 303), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`
 (p. 343), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`
 (p. 369), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`
 (p. 413), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`
 (p. 455), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`
 (p. 514), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`
 (p. 539), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`
 (p. 567), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`
 (p. 595), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`
 (p. 623), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`
 (p. 651), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`
 (p. 723), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 815),
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 846), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1191),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1222),
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1252),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1282),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1325),
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1352),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1384), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1411), `ac-`
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1444),
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1482),
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1640),
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1669), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1737),
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1837),
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1981),
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2027),
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2048),
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2094), `ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2113),
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2143),
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`
 (p. 2172), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`
 (p. 2218), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`
 (p. 2421), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`
 (p. 2448), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`
 (p. 2489), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2509),
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2525), `ac-`
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2574), `ac-`
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2627),
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2742),
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2857),
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 2888),
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2916),
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3009), `ac-`
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`
 (p. 3037), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`
 (p. 3068), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3096),
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3179), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3198), `ac-`

tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3266),
 tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3448),
 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (p. 3574),
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3601),
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3730),
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3791),
 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller
 (p. 193), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller
 (p. 228), activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller
 (p. 311), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller
 (p. 351), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller
 (p. 377), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller
 (p. 421), activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller
 (p. 463), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller
 (p. 522), activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller
 (p. 547), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller
 (p. 575), activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller
 (p. 603), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller
 (p. 627), activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller
 (p. 655), activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller
 (p. 730), activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 819),
 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 850),
 tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1195),
 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1226),
 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1256),
 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1286),
 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1329),
 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1356),
 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1388),
 tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1415),
 tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1448),
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1486),
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1636),
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1649),
 tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1725),
 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1817),
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 1961),
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2019),
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2060),
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2082),
 tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2101),
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2127),
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller
 (p. 2160), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller
 (p. 2206), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller
 (p. 2401), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller
 (p. 2460), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller
 (p. 2469), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2517),
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2545),
 tivemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2590),
 tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2623),
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 2733),
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 2861),
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 2892),

`activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 2924),
`activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 2997), `ac-`
`tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller`
 (p. 3033), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller`
 (p. 3064), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109),
`activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3175), `ac-`
`tivemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3194), `ac-`
`tivemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3254), `ac-`
`tivemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3456),
`activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3592), `ac-`
`tivemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3613), `ac-`
`tivemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 3734), and
`activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 3771).

6.301.3.5 `virtual int ac-`
`tivemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal1`
 (`OpenWireFormat * format`, `commands::DataStructure * command`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [pure
 virtual]

Tight Marshal to the given stream.

Parameters

format - The OpenWireFormat properties

command - the object to Marshal

bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller`
 (p. 177), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller`
 (p. 216), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller`
 (p. 296), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller`
 (p. 335), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller`
 (p. 361), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller`
 (p. 405), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller`
 (p. 447), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller`
 (p. 506), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller`
 (p. 532), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller`
 (p. 559), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller`
 (p. 591), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`
 (p. 619), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller`
 (p. 647), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`
 (p. 718), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 811),
`activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 842), `ac-`
`tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1187),
`activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1218),
`activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1248),
`activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1278),
`activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1321),

activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1348),
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1380), ac-
 tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1407), ac-
 tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1440),
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1502),
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1632),
 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1665), ac-
 tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1745),
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1833),
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1977),
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 2039),
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 2068),
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2090), ac-
 tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2121),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2147),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 2181), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 2226), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2417), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2456), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2485), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2521),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2542), ac-
 tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2586), ac-
 tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2639),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2755),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2865),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2896),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2912),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3005), ac-
 tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 3022), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 3052), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3105),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3187), ac-
 tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3202), ac-
 tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3262), ac-
 tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3445),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3578),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3605),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3746),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3783),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 185), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 232), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 308), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 347), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 373), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 417), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 459), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 518), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 543), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 571), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 599), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 631), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 659), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller

(p. 738), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 823),
`activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 854), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1199),
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1207),
`activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1236),
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1266),
`activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1309),
`activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1336),
`activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1368), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1396), `ac-`
`tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1428),
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1490),
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1621),
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1653), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1729),
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1821),
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1965),
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2023),
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2052),
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2074), `ac-`
`tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2105),
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2131),
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`
(p. 2169), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller`
(p. 2210), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller`
(p. 2405), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`
(p. 2440), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`
(p. 2473), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2501),
`activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2534), `ac-`
`tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2571), `ac-`
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2619),
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2738),
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2845),
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 2876),
`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 2908),
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2994), `ac-`
`tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`
(p. 3029), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`
(p. 3056), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3092),
`activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 3167), `ac-`
`tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3210), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3258), `ac-`
`tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3460),
`activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3582),
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3621),
`activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller` (p. 3738),
`activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3775),
`activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`
(p. 173), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`
(p. 212), `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller`
(p. 292), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`
(p. 331), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`
(p. 357), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`
(p. 401), `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller`

(p. 443), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`
 (p. 502), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller`
 (p. 529), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller`
 (p. 555), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller`
 (p. 583), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`
 (p. 612), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`
 (p. 639), `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`
 (p. 704), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 803),
`activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 834), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1179),
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1210),
`activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1240),
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1270),
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1313),
`activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1340),
`activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1372), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1399), `ac-`
`tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1432),
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1494),
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1624),
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1657), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1733),
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1825),
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1969),
`activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2031),
`activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2056),
`activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2078), `ac-`
`tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2109),
`activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2135),
`activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller`
 (p. 2165), `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller`
 (p. 2214), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller`
 (p. 2409), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller`
 (p. 2444), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`
 (p. 2477), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2513),
`activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2530), `ac-`
`tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2578), `ac-`
`tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2631),
`activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2746),
`activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2853),
`activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 2884),
`activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 2920),
`activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3001), `ac-`
`tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller`
 (p. 3025), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`
 (p. 3060), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3101),
`activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3183), `ac-`
`tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3206), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3270), `ac-`
`tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3441),
`activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3586),
`activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3609),
`activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 3750),
`activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3787),

activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 181), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 220), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
 (p. 300), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 339), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 365), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 409), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 451), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 510), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
 (p. 536), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 563), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
 (p. 587), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
 (p. 615), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
 (p. 643), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller
 (p. 711), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 807),
 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 838), ac-
 tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1183),
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1214),
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1244),
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1274),
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1317),
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1344),
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1376), ac-
 tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1403), ac-
 tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1436),
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1498),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1628),
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1661), ac-
 tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1741),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1829),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1973),
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2035),
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2064),
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2086), ac-
 tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2117),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2139),
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller
 (p. 2177), activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller
 (p. 2222), activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller
 (p. 2413), activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller
 (p. 2452), activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller
 (p. 2481), activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2505),
 activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2538), ac-
 tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2582), ac-
 tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2635),
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2750),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2849),
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 2880),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2904),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3013), ac-
 tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller
 (p. 3041), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller
 (p. 3048), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3087),
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3171), ac-

tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3214), ac-
 tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3274), ac-
 tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3452),
 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3589),
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3617),
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3742),
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3779),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller
 (p. 189), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller
 (p. 224), activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller
 (p. 304), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller
 (p. 343), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller
 (p. 369), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller
 (p. 413), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller
 (p. 455), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller
 (p. 514), activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller
 (p. 540), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller
 (p. 567), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller
 (p. 595), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller
 (p. 623), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller
 (p. 651), activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller
 (p. 724), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 815),
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 846), ac-
 tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1191),
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1222),
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1252),
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1282),
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1325),
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1352),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1384), ac-
 tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1411), ac-
 tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1444),
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1482),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1640),
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1669), ac-
 tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1737),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1837),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1981),
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2027),
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2048),
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2094), ac-
 tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2113),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2143),
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
 (p. 2173), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller
 (p. 2218), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
 (p. 2421), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
 (p. 2448), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2489), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2509),
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2526), ac-
 tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2574), ac-
 tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2627),
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2742),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2857),

activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 2888),
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2916),
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3009), ac-
tivismq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
(p. 3037), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
(p. 3068), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3096),
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3179), ac-
tivismq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3198), ac-
tivismq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3266), ac-
tivismq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3448),
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (p. 3575),
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3601),
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3730),
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3791),
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller
(p. 193), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller
(p. 228), activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller
(p. 312), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller
(p. 351), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller
(p. 377), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller
(p. 421), activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller
(p. 463), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller
(p. 522), activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller
(p. 547), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller
(p. 575), activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller
(p. 603), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller
(p. 627), activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller
(p. 655), activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller
(p. 731), activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 819),
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 850), ac-
tivismq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1195),
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1226),
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1256),
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1286),
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1329),
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1356),
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1388), ac-
tivismq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1415), ac-
tivismq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1448),
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1486),
activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1636),
activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1649), ac-
tivismq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1725),
activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1817),
activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 1961),
activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2020),
activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2060),
activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2082), ac-
tivismq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2101),
activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2128),
activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller
(p. 2161), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller
(p. 2206), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller
(p. 2402), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller

(p. 2460), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2469), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2517), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2546), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2590), `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2623), `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2734), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 2861), `activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 2892), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 2924), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 2997), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3033), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3064), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3110), `activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3175), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3194), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3255), `activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3456), `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3593), `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3613), `activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 3734), and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 3771).

6.301.3.6 `virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) throw (decaf::io::IOException)` [pure virtual]

Tight Marshal to the given stream.

Parameters

format - The OpenWireFormat properties

command - the object to Marshal

ds - the DataOutputStream to Marshal to

bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 177), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 216), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 296), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 335), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 362), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 406), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 448), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 506), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 533), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 559), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 591), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`

(p. 620), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller`
 (p. 647), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`
 (p. 719), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 811),
`activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 842), `ac-`
`tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1187),
`activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1219),
`activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1248),
`activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1278),
`activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1321),
`activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1348),
`activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1380), `ac-`
`tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1408), `ac-`
`tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1441),
`activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1503),
`activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1633),
`activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1665), `ac-`
`tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1746),
`activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1833),
`activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1978),
`activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2039),
`activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2068),
`activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2090), `ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2121),
`activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2148),
`activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller`
 (p. 2181), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller`
 (p. 2227), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller`
 (p. 2418), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller`
 (p. 2457), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller`
 (p. 2485), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2521),
`activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543), `ac-`
`tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2587), `ac-`
`tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2639),
`activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2755),
`activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2865),
`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 2896),
`activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2913),
`activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3006), `ac-`
`tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller`
 (p. 3022), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller`
 (p. 3053), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3106),
`activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3187), `ac-`
`tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3202), `ac-`
`tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3263), `ac-`
`tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3445),
`activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3579),
`activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3605),
`activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller` (p. 3746),
`activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3784),
`activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`
 (p. 185), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller`
 (p. 232), `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`
 (p. 308), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`
 (p. 347), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`

(p. 374), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`
 (p. 418), `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller`
 (p. 460), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`
 (p. 518), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller`
 (p. 544), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller`
 (p. 571), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller`
 (p. 599), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`
 (p. 632), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller`
 (p. 659), `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`
 (p. 739), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 823),
`activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 854), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1199),
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1207),
`activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1237),
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1266),
`activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1309),
`activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1337),
`activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1369), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1396), `ac-`
`tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1429),
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1491),
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1621),
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1654), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1730),
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1821),
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1966),
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2024),
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2052),
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2075), `ac-`
`tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2105),
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2132),
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`
 (p. 2169), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller`
 (p. 2211), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller`
 (p. 2406), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`
 (p. 2441), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`
 (p. 2473), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2502),
`activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2535), `ac-`
`tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2571), `ac-`
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2620),
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2738),
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2845),
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 2877),
`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 2909),
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2994), `ac-`
`tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`
 (p. 3030), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`
 (p. 3057), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3092),
`activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 3168), `ac-`
`tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3210), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3259), `ac-`
`tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3460),
`activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3582),
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3621),

activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3738),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3776),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 174), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 213), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
 (p. 292), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 331), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 358), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 402), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 444), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 502), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller
 (p. 529), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 555), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 583), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 612), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 639), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
 (p. 706), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 804),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 834), ac-
 tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1180),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1211),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1241),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1270),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1313),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1341),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1372), ac-
 tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1400), ac-
 tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1433),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1495),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1625),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1657), ac-
 tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1734),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1825),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1970),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2031),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2056),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2079), ac-
 tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2109),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2136),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 2165), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 2215), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2410), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2445), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2477), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2513),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2531), ac-
 tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2579), ac-
 tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2631),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2747),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2853),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2884),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2921),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3002), ac-
 tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 3026), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller

(p. 3061), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3101),
`activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3183), `ac-`
`tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3206), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3271), `ac-`
`tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3441),
`activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3586),
`activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3609),
`activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 3750),
`activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3788),
`activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`
(p. 181), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`
(p. 220), `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller`
(p. 300), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`
(p. 339), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`
(p. 366), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`
(p. 410), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`
(p. 452), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`
(p. 510), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller`
(p. 537), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`
(p. 563), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`
(p. 587), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`
(p. 616), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`
(p. 643), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`
(p. 712), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 808),
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 838), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1183),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1215),
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1244),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1274),
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1317),
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1345),
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1376), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1404), `ac-`
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1437),
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1499),
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1629),
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1661), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1742),
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1829),
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1974),
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2035),
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2064),
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2086), `ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2117),
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2140),
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`
(p. 2177), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`
(p. 2223), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`
(p. 2414), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`
(p. 2453), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`
(p. 2481), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2506),
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2539), `ac-`
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2583), `ac-`
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2635),

activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2751),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2849),
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 2881),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2905),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3014),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller
 (p. 3042),
 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller
 (p. 3049),
 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3088),
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3172),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3214),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3275),
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3453),
 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3590),
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3617),
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3742),
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3780),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller
 (p. 189),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller
 (p. 224),
 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller
 (p. 304),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller
 (p. 343),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller
 (p. 370),
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller
 (p. 414),
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller
 (p. 456),
 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller
 (p. 514),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller
 (p. 540),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller
 (p. 567),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller
 (p. 595),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller
 (p. 624),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller
 (p. 651),
 activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller
 (p. 726),
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 815),
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 846),
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1191),
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1223),
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1252),
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1282),
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1325),
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1352),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1384),
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1412),
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1445),
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1483),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1641),
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1669),
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1738),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1837),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1982),
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2028),
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2048),
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2094),
 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2113),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2144),
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
 (p. 2173),
 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller

(p. 2219), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
 (p. 2422), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
 (p. 2449), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2489), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2510),
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2526), ac-
 tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2575), ac-
 tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2627),
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2743),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2857),
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 2888),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2917),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3010), ac-
 tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
 (p. 3038), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
 (p. 3069), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3097),
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3179), ac-
 tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3198), ac-
 tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3267), ac-
 tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3449),
 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (p. 3575),
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3601),
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3731),
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3792),
 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller
 (p. 193), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller
 (p. 228), activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller
 (p. 312), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller
 (p. 351), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller
 (p. 378), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller
 (p. 422), activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller
 (p. 464), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller
 (p. 522), activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller
 (p. 548), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller
 (p. 575), activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller
 (p. 603), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller
 (p. 628), activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller
 (p. 655), activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller
 (p. 732), activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 819),
 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 850), ac-
 tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1195),
 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1227),
 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1256),
 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1286),
 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1329),
 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1356),
 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1388), ac-
 tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1416), ac-
 tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1449),
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1487),
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1637),
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1650), ac-
 tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1726),
 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1817),
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 1962),

activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2020),
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2060),
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2082),
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2102),
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2128),
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller
 (p. 2161), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller
 (p. 2207), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller
 (p. 2402), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller
 (p. 2461), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller
 (p. 2469), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2517),
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2547),
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2591),
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2623),
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 2734),
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 2861),
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 2892),
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 2925),
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 2998),
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller
 (p. 3034), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller
 (p. 3065), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3110),
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (p. 3176),
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3194),
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3255),
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3456),
 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller (p. 3593),
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3613),
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 3734), and
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 3772).

6.301.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::DataStreamMarshaller::tightUnmarshal
(OpenWireFormat * *format*, commands::DataStructure * *command*,
decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) throw (
decaf::io::IOException) [pure virtual]

Tight Un-marhsal to the given stream.

Parameters

format - The OpenWireFormat properties
command - the object to Un-Marshal
dis - the DataInputStream to Un-Marshal from
bs - boolean stream to unmarshal from.

Exceptions

IOException if an error occurs.

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller`
 (p. 178), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller`
 (p. 217), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller`

(p. 297), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller`
 (p. 336), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller`
 (p. 362), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller`
 (p. 406), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller`
 (p. 448), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller`
 (p. 507), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller`
 (p. 533), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller`
 (p. 560), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller`
 (p. 592), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`
 (p. 620), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller`
 (p. 648), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`
 (p. 720), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 812),
`activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 843), `ac-`
`tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1188),
`activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1219),
`activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1249),
`activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1279),
`activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1322),
`activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1349),
`activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1381), `ac-`
`tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1408), `ac-`
`tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1441),
`activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1503),
`activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1633),
`activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1666), `ac-`
`tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1746),
`activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1834),
`activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1978),
`activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2040),
`activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2069),
`activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2091), `ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2122),
`activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2148),
`activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller`
 (p. 2181), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller`
 (p. 2227), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller`
 (p. 2418), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller`
 (p. 2457), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller`
 (p. 2486), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2522),
`activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2543), `ac-`
`tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2587), `ac-`
`tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2640),
`activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2756),
`activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2866),
`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 2897),
`activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2913),
`activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3006), `ac-`
`tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller`
 (p. 3022), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller`
 (p. 3053), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3106),
`activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3188), `ac-`
`tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3203), `ac-`
`tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3263), `ac-`
`tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3445),

activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3579),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3606),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3747),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3784),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 186), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 233), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 309), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 348), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 374), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 418), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 460), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 519), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 544), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 572), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 600), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 632), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 660), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 740), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 824),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 855), ac-
 tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1200),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1207),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1237),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1267),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1310),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1337),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1369), ac-
 tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1396), ac-
 tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1429),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1491),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1621),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1654), ac-
 tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1730),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1822),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1966),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2024),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2053),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2075), ac-
 tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2106),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2132),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
 (p. 2169), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller
 (p. 2211), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
 (p. 2406), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
 (p. 2441), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2474), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2502),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2535), ac-
 tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2571), ac-
 tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2620),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2739),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2846),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2877),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2909),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2994), ac-

tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 3030), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 3057), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3093),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3168), ac-
 tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3211), ac-
 tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3259), ac-
 tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3461),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3583),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3622),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3739),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3776),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 174), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 213), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
 (p. 293), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 332), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 358), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 402), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 444), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 503), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller
 (p. 530), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 556), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 584), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 612), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 640), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
 (p. 707), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 804),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 835), ac-
 tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1180),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1211),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1241),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1271),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1314),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1341),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1373), ac-
 tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1400), ac-
 tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1433),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1495),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1625),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1658), ac-
 tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1734),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1826),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1970),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2032),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2057),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2079), ac-
 tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2110),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2136),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 2165), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 2215), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2410), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2445), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2478), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2514),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2531), ac-

activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2579),
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2632),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2747),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2854),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2885),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2921),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3002),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 3026),
 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 3061),
 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3102),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3184),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3207),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3271),
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3442),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3587),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3610),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3751),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3788),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 182),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 221),
 activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
 (p. 301),
 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 340),
 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 366),
 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 410),
 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 452),
 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 511),
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
 (p. 537),
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 564),
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
 (p. 588),
 activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
 (p. 616),
 activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
 (p. 644),
 activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller
 (p. 713),
 activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 808),
 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 839),
 activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1184),
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1215),
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1245),
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1275),
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1318),
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1345),
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1377),
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1404),
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1437),
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1499),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1629),
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1662),
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1742),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1830),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1974),
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2036),
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2065),
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2087),
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2118),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2140),

activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (p. 2177),
 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (p. 2223),
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2414),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2453),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (p. 2482),
 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2506),
 activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2539),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2583),
 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2636),
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2751),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2850),
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 2881),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2905),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3014),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (p. 3042),
 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (p. 3049),
 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3088),
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3172),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3215),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3275),
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3453),
 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3590),
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3618),
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3743),
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3780),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 190),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (p. 225),
 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller (p. 305),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (p. 344),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (p. 370),
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (p. 414),
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (p. 456),
 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller (p. 515),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller (p. 541),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller (p. 568),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller (p. 596),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller (p. 624),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller (p. 652),
 activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller (p. 727),
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 816),
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 847),
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1192),
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1223),
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1253),
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1283),
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1326),
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1353),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1385),
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1412),
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1445),
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1483),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1641),
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1670),
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1738),

activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1838),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1982),
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2028),
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2049),
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2095),
 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2114),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2144),
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
 (p. 2173), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller
 (p. 2219), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
 (p. 2422), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
 (p. 2449), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2490), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2510),
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2527),
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2575),
 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2628),
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2743),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2858),
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 2889),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2917),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3010),
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
 (p. 3038), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
 (p. 3069), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3097),
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3180),
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3199),
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3267),
 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3449),
 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (p. 3576),
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3602),
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3731),
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3792),
 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller
 (p. 194), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller
 (p. 229), activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller
 (p. 313), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller
 (p. 352), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller
 (p. 378), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller
 (p. 422), activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller
 (p. 464), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller
 (p. 523), activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller
 (p. 548), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller
 (p. 576), activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller
 (p. 604), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller
 (p. 628), activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller
 (p. 656), activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller
 (p. 733), activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 820),
 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 851),
 activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1196),
 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1227),
 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1257),
 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1287),
 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1330),
 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1357),

activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1389),
 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1416),
 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1449),
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1487),
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1637),
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1650),
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1726),
 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1818),
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 1962),
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2020),
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2061),
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2083),
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2102),
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2128),
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller
 (p. 2162), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller
 (p. 2207), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller
 (p. 2402), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller
 (p. 2461), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller
 (p. 2470), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2518),
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2547),
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2591),
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2624),
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 2735),
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 2862),
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 2893),
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 2925),
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 2998),
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller
 (p. 3034), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller
 (p. 3065), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3111),
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (p. 3176),
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3195),
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3255),
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3457),
 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller (p. 3594),
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3614),
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 3735), and
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 3772).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h`

6.302 activemq::commands::DataStructure Class Reference

```
#include <src/main/activemq/commands/DataStructure.h>
```

Inheritance diagram for activemq::commands::DataStructure:

Public Member Functions

- virtual `~DataStructure ()`
- virtual unsigned char `getDataStructureType ()` const =0
*Get the **DataStructure** (p. 1553) Type as defined in *CommandTypes.h*.*
- virtual `DataStructure * cloneDataStructure ()` const =0
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void `copyDataStructure (const DataStructure *src)=0`
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string `toString ()` const =0
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool `equals (const DataStructure *value) const =0`
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

6.302.1 Constructor & Destructor Documentation

- 6.302.1.1** virtual `activemq::commands::DataStructure::~DataStructure ()`
 [inline, virtual]

6.302.2 Member Function Documentation

- 6.302.2.1** virtual `DataStructure* activemq::commands::DataStructure::cloneDataStructure ()` const [pure virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 168), `activemq::commands::ActiveMQBytesMessage` (p. 198), `activemq::commands::ActiveMQDestination` (p. 282), `activemq::commands::ActiveMQMapMessage` (p. 320), `activemq::commands::ActiveMQMessage` (p. 354), `activemq::commands::ActiveMQObjectMessage` (p. 397), `activemq::commands::ActiveMQStreamMessage` (p. 489), `activemq::commands::ActiveMQTextMessage` (p. 607), `activemq::commands::BooleanExpression` (p. 786), `activemq::commands::BrokerError` (p. 794), `activemq::commands::BrokerInfo` (p. 826), `activemq::commands::ConnectionControl` (p. 1173), `activemq::commands::ConnectionError` (p. 1202), `activemq::commands::ConnectionInfo` (p. 1259), `activemq::commands::ConsumerControl` (p. 1303), `activemq::commands::ConsumerInfo` (p. 1360), `activemq::commands::ControlCommand` (p. 1391),

activemq::commands::DataArrayResponse (p. 1424), **activemq::commands::DataResponse** (p. 1478), **activemq::commands::DestinationInfo** (p. 1615), **activemq::commands::DiscoveryEvent** (p. 1645), **activemq::commands::ExceptionResponse** (p. 1721), **activemq::commands::FlushCommand** (p. 1813), **activemq::commands::IntegerResponse** (p. 1957), **activemq::commands::JournalQueueAck** (p. 2015), **activemq::commands::JournalTopicAck** (p. 2042), **activemq::commands::JournalTrace** (p. 2070), **activemq::commands::JournalTransaction** (p. 2096), **activemq::commands::KeepAliveInfo** (p. 2123), **activemq::commands::LastPartialCommand** (p. 2157), **activemq::commands::Message** (p. 2363), **activemq::commands::MessageAck** (p. 2395), **activemq::commands::MessageDispatch** (p. 2428), **activemq::commands::MessageDispatchNotification** (p. 2463), **activemq::commands::MessagePull** (p. 2565), **activemq::commands::NetworkBridgeFilter** (p. 2615), **activemq::commands::PartialCommand** (p. 2729), **activemq::commands::ProducerAck** (p. 2840), **activemq::commands::ProducerInfo** (p. 2899), **activemq::commands::RemoveInfo** (p. 2989), **activemq::commands::RemoveSubscriptionInfo** (p. 3016), **activemq::commands::ReplayCommand** (p. 3044), **activemq::commands::Response** (p. 3077), **activemq::commands::SessionInfo** (p. 3189), **activemq::commands::ShutdownInfo** (p. 3250), **activemq::commands::SubscriptionInfo** (p. 3435), **activemq::commands::TransactionInfo** (p. 3596), and **activemq::commands::WireFormatInfo** (p. 3721).

6.302.2.2 virtual void activemq::commands::DataStructure::copyDataStructure (const DataStructure * src) [pure virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 168), **activemq::commands::ActiveMQBytesMessage** (p. 198), **activemq::commands::ActiveMQDestination** (p. 282), **activemq::commands::ActiveMQMapMessage** (p. 320), **activemq::commands::ActiveMQMessage** (p. 354), **activemq::commands::ActiveMQObjectMessage** (p. 398), **activemq::commands::ActiveMQStreamMessage** (p. 489), **activemq::commands::ActiveMQTextMessage** (p. 607), **activemq::commands::BaseCommand** (p. 696), **activemq::commands::BrokerError** (p. 794), **activemq::commands::BrokerInfo** (p. 826), **activemq::commands::ConnectionControl** (p. 1173), **activemq::commands::ConnectionError** (p. 1202), **activemq::commands::ConnectionInfo** (p. 1259), **activemq::commands::ConsumerControl** (p. 1304), **activemq::commands::ConsumerInfo** (p. 1360), **activemq::commands::ControlCommand** (p. 1391), **activemq::commands::DataArrayResponse** (p. 1424), **activemq::commands::DataResponse** (p. 1478), **activemq::commands::DestinationInfo** (p. 1615), **activemq::commands::DiscoveryEvent** (p. 1645), **activemq::commands::ExceptionResponse** (p. 1721), **activemq::commands::FlushCommand** (p. 1813), **activemq::commands::IntegerResponse** (p. 1957), **activemq::commands::JournalQueueAck** (p. 2015), **activemq::commands::JournalTopicAck** (p. 2042), **activemq::commands::JournalTrace** (p. 2071), **activemq::commands::JournalTransaction** (p. 2097), **activemq::commands::KeepAliveInfo** (p. 2124), **activemq::commands::LastPartialCommand**

(p. 2157), `activemq::commands::Message` (p. 2363), `activemq::commands::MessageAck` (p. 2395), `activemq::commands::MessageDispatch` (p. 2428), `activemq::commands::MessageDispatchNotification` (p. 2463), `activemq::commands::MessagePull` (p. 2565), `activemq::commands::NetworkBridgeFilter` (p. 2615), `activemq::commands::PartialCommand` (p. 2729), `activemq::commands::ProducerAck` (p. 2841), `activemq::commands::ProducerInfo` (p. 2899), `activemq::commands::RemoveInfo` (p. 2989), `activemq::commands::RemoveSubscriptionInfo` (p. 3016), `activemq::commands::ReplayCommand` (p. 3044), `activemq::commands::Response` (p. 3078), `activemq::commands::SessionInfo` (p. 3190), `activemq::commands::ShutdownInfo` (p. 3251), `activemq::commands::SubscriptionInfo` (p. 3435), `activemq::commands::TransactionInfo` (p. 3596), and `activemq::commands::WireFormatInfo` (p. 3721).

6.302.2.3 `virtual bool activemq::commands::DataStructure::equals (const DataStructure * value) const` [pure virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 168), `activemq::commands::ActiveMQBytesMessage` (p. 199), `activemq::commands::ActiveMQDestination` (p. 283), `activemq::commands::ActiveMQMapMessage` (p. 320), `activemq::commands::ActiveMQMessage` (p. 354), `activemq::commands::ActiveMQMessageTemplate< T >` (p. 383), `activemq::commands::ActiveMQObjectMessage` (p. 398), `activemq::commands::ActiveMQStreamMessage` (p. 490), `activemq::commands::ActiveMQTextMessage` (p. 607), `activemq::commands::BaseCommand` (p. 696), `activemq::commands::BooleanExpression` (p. 787), `activemq::commands::BrokerInfo` (p. 827), `activemq::commands::ConnectionControl` (p. 1174), `activemq::commands::ConnectionError` (p. 1202), `activemq::commands::ConnectionInfo` (p. 1260), `activemq::commands::ConsumerControl` (p. 1304), `activemq::commands::ConsumerInfo` (p. 1360), `activemq::commands::ControlCommand` (p. 1392), `activemq::commands::DataArrayResponse` (p. 1424), `activemq::commands::DataResponse` (p. 1478), `activemq::commands::DestinationInfo` (p. 1615), `activemq::commands::DiscoveryEvent` (p. 1645), `activemq::commands::ExceptionResponse` (p. 1721), `activemq::commands::FlushCommand` (p. 1813), `activemq::commands::IntegerResponse` (p. 1957), `activemq::commands::JournalQueueAck` (p. 2016), `activemq::commands::JournalTopicAck` (p. 2042), `activemq::commands::JournalTrace` (p. 2071), `activemq::commands::JournalTransaction` (p. 2097), `activemq::commands::KeepAliveInfo` (p. 2124), `activemq::commands::LastPartialCommand` (p. 2157), `activemq::commands::Message` (p. 2363), `activemq::commands::MessageAck` (p. 2396), `activemq::commands::MessageDispatch` (p. 2428), `activemq::commands::MessageDispatchNotification` (p. 2464), `activemq::commands::MessagePull` (p. 2565), `activemq::commands::NetworkBridgeFilter` (p. 2615), `activemq::commands::PartialCommand` (p. 2729), `activemq::commands::ProducerAck` (p. 2841), `activemq::commands::ProducerInfo`

(p. 2899), **activemq::commands::RemoveInfo** (p. 2989), **ac-**
tivemq::commands::RemoveSubscriptionInfo (p. 3017), **ac-**
tivemq::commands::ReplayCommand (p. 3044), **activemq::commands::Response**
 (p. 3078), **activemq::commands::SessionInfo** (p. 3190), **ac-**
tivemq::commands::ShutdownInfo (p. 3251), **activemq::commands::SubscriptionInfo**
 (p. 3435), **activemq::commands::TransactionInfo** (p. 3596), **ac-**
tivemq::commands::WireFormatInfo (p. 3721), **activemq::commands::ActiveMQMessageTemplate<**
cms::BytesMessage > (p. 383), **activemq::commands::ActiveMQMessageTemplate<**
cms::MapMessage > (p. 383), **activemq::commands::ActiveMQMessageTemplate<**
cms::Message > (p. 383), **activemq::commands::ActiveMQMessageTemplate<**
cms::StreamMessage > (p. 383), **activemq::commands::ActiveMQMessageTemplate<**
cms::TextMessage > (p. 383), and **activemq::commands::ActiveMQMessageTemplate<**
cms::ObjectMessage > (p. 383).

6.302.2.4 virtual unsigned char **activemq::commands::DataStructure::getDataStructureType** () const [pure virtual]

Get the **DataStructure** (p. 1553) Type as defined in **CommandTypes.h**.

Returns

The type of the data structure

Implemented in **activemq::commands::ActiveMQBlobMessage**
 (p. 168), **activemq::commands::ActiveMQBytesMessage** (p. 200),
activemq::commands::ActiveMQDestination (p. 284), **ac-**
tivemq::commands::ActiveMQMapMessage (p. 322), **ac-**
tivemq::commands::ActiveMQMessage (p. 354), **activemq::commands::ActiveMQObjectMessage**
 (p. 398), **activemq::commands::ActiveMQStreamMessage** (p. 490),
activemq::commands::ActiveMQTextMessage (p. 607), **ac-**
tivemq::commands::BrokerError (p. 795), **activemq::commands::BrokerInfo**
 (p. 827), **activemq::commands::ConnectionControl** (p. 1174), **ac-**
tivemq::commands::ConnectionError (p. 1203), **activemq::commands::ConnectionInfo**
 (p. 1260), **activemq::commands::ConsumerControl** (p. 1304), **ac-**
tivemq::commands::ConsumerInfo (p. 1361), **activemq::commands::ControlCommand**
 (p. 1392), **activemq::commands::DataArrayResponse** (p. 1425), **ac-**
tivemq::commands::DataResponse (p. 1479), **activemq::commands::DestinationInfo**
 (p. 1616), **activemq::commands::DiscoveryEvent** (p. 1646),
activemq::commands::ExceptionResponse (p. 1721), **ac-**
tivemq::commands::FlushCommand (p. 1813), **activemq::commands::IntegerResponse**
 (p. 1958), **activemq::commands::JournalQueueAck** (p. 2016), **ac-**
tivemq::commands::JournalTopicAck (p. 2043), **activemq::commands::JournalTrace**
 (p. 2071), **activemq::commands::JournalTransaction** (p. 2097), **ac-**
tivemq::commands::KeepAliveInfo (p. 2124), **activemq::commands::LastPartialCommand**
 (p. 2157), **activemq::commands::Message** (p. 2365), **activemq::commands::MessageAck**
 (p. 2396), **activemq::commands::MessageDispatch** (p. 2429), **ac-**
tivemq::commands::MessageDispatchNotification (p. 2464), **ac-**
tivemq::commands::MessagePull (p. 2566), **activemq::commands::NetworkBridgeFilter**
 (p. 2615), **activemq::commands::PartialCommand** (p. 2730), **ac-**
tivemq::commands::ProducerAck (p. 2841), **activemq::commands::ProducerInfo**
 (p. 2900), **activemq::commands::RemoveInfo** (p. 2990), **ac-**
tivemq::commands::RemoveSubscriptionInfo (p. 3017), **ac-**
tivemq::commands::ReplayCommand (p. 3045), **activemq::commands::Response**

(p. 3078), `activemq::commands::SessionInfo` (p. 3190), `activemq::commands::ShutdownInfo` (p. 3251), `activemq::commands::SubscriptionInfo` (p. 3436), `activemq::commands::TransactionInfo` (p. 3597), and `activemq::commands::WireFormatInfo` (p. 3721).

6.302.2.5 `virtual std::string activemq::commands::DataStructure::toString ()`
`const [pure virtual]`

Returns a string containing the information for this `DataStructure` (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 170), `activemq::commands::ActiveMQBytesMessage` (p. 205), `activemq::commands::ActiveMQDestination` (p. 288), `activemq::commands::ActiveMQMapMessage` (p. 328), `activemq::commands::ActiveMQMessage` (p. 355), `activemq::commands::ActiveMQObjectMessage` (p. 398), `activemq::commands::ActiveMQStreamMessage` (p. 495), `activemq::commands::ActiveMQTextMessage` (p. 609), `activemq::commands::BaseCommand` (p. 700), `activemq::commands::BaseDataStructure` (p. 767), `activemq::commands::BooleanExpression` (p. 787), `activemq::commands::BrokerInfo` (p. 829), `activemq::commands::Command` (p. 1111), `activemq::commands::ConnectionControl` (p. 1175), `activemq::commands::ConnectionError` (p. 1203), `activemq::commands::ConnectionInfo` (p. 1262), `activemq::commands::ConsumerControl` (p. 1305), `activemq::commands::ConsumerInfo` (p. 1363), `activemq::commands::ControlCommand` (p. 1392), `activemq::commands::DataArrayResponse` (p. 1425), `activemq::commands::DataResponse` (p. 1479), `activemq::commands::DestinationInfo` (p. 1617), `activemq::commands::DiscoveryEvent` (p. 1646), `activemq::commands::ExceptionResponse` (p. 1722), `activemq::commands::FlushCommand` (p. 1814), `activemq::commands::IntegerResponse` (p. 1958), `activemq::commands::JournalQueueAck` (p. 2016), `activemq::commands::JournalTopicAck` (p. 2044), `activemq::commands::JournalTrace` (p. 2071), `activemq::commands::JournalTransaction` (p. 2098), `activemq::commands::KeepAliveInfo` (p. 2124), `activemq::commands::LastPartialCommand` (p. 2158), `activemq::commands::Message` (p. 2372), `activemq::commands::MessageAck` (p. 2398), `activemq::commands::MessageDispatch` (p. 2430), `activemq::commands::MessageDispatchNotification` (p. 2465), `activemq::commands::MessagePull` (p. 2567), `activemq::commands::NetworkBridgeFilter` (p. 2616), `activemq::commands::PartialCommand` (p. 2730), `activemq::commands::ProducerAck` (p. 2842), `activemq::commands::ProducerInfo` (p. 2901), `activemq::commands::RemoveInfo` (p. 2990), `activemq::commands::RemoveSubscriptionInfo` (p. 3018), `activemq::commands::ReplayCommand` (p. 3045), `activemq::commands::Response` (p. 3079), `activemq::commands::SessionInfo` (p. 3191), `activemq::commands::ShutdownInfo` (p. 3251), `activemq::commands::SubscriptionInfo` (p. 3437), `activemq::commands::TransactionInfo` (p. 3597), and `activemq::commands::WireFormatInfo` (p. 3727).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataStructure.h`

6.303 decaf::util::Date Class Reference

Wrapper class around a time value in milliseconds.

```
#include <src/main/decaf/util/Date.h>
```

Inheritance diagram for decaf::util::Date:

Public Member Functions

- **Date** ()
Default constructor - sets time to the current System time, rounded to the nearest millisecond.
- **Date** (long long milliseconds)
Constructs the date with a given time value.
- **Date** (const **Date** &source)
Copy constructor.
- **Date** & **operator=** (const **Date** &value)
*Assigns the value of one **Date** (p. 1559) object to another.*
- virtual ~**Date** ()
- long long **getTime** () const
Gets the underlying time.
- void **setTime** (long long milliseconds)
Sets the underlying time.
- bool **after** (const **Date** &when) const
Determines whether or not this date falls after the specified time.
- bool **before** (const **Date** &when) const
Determines whether or not this date falls before the specified time.
- std::string **toString** () const
*Converts this **Date** (p. 1559) object to a String of the form:*
- virtual int **compareTo** (const **Date** &value) const
Compares this Data object to the one given.
- virtual bool **equals** (const **Date** &value) const
- virtual bool **operator==** (const **Date** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Date** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

6.303.1 Detailed Description

Wrapper class around a time value in milliseconds. This class is comparable to Java's `java.util.Date` class.

Since

1.0

6.303.2 Constructor & Destructor Documentation

6.303.2.1 `decaf::util::Date::Date ()`

Default constructor - sets time to the current System time, rounded to the nearest millisecond.

6.303.2.2 `decaf::util::Date::Date (long long milliseconds)`

Constructs the date with a given time value.

Parameters

milliseconds The time in milliseconds;

6.303.2.3 `decaf::util::Date::Date (const Date & source)`

Copy constructor.

Parameters

source The `Date` (p. 1559) instance to copy into this one.

6.303.2.4 `virtual decaf::util::Date::~~Date ()` [virtual]

6.303.3 Member Function Documentation

6.303.3.1 `bool decaf::util::Date::after (const Date & when) const`

Determines whether or not this date falls after the specified time.

Parameters

when The date to compare

Returns

true if this date falls after when.

6.303.3.2 `bool decaf::util::Date::before (const Date & when) const`

Determines whether or not this date falls before the specified time.

Parameters

when The date to compare

Returns

true if this date falls before when.

6.303.3.3 `virtual int decaf::util::Date::compareTo (const Date & value) const`
[virtual]

Compares this Date object to the one given.

Parameters

value The **Date** (p. 1559) value to compare to this one.

Returns

zero if the **Date** (p. 1559) values are equal, a value less than zero if this Date value is earlier than argument value, and a value greater than zero if this **Date** (p. 1559) object is later than the argument **Date** (p. 1559) value.

6.303.3.4 `virtual bool decaf::util::Date::equals (const Date & value) const`
[virtual]**Returns**

true if this value is considered equal to the passed value.

6.303.3.5 `long long decaf::util::Date::getTime () const`

Gets the underlying time.

Returns

The underlying time value in milliseconds.

6.303.3.6 `virtual bool decaf::util::Date::operator< (const Date & value) const`
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.303.3.7 Date& decaf::util::Date::operator= (const Date & value)

Assigns the value of one **Date** (p. 1559) object to another.

Parameters

value The value to be copied into this **Date** (p. 1559) object.

Returns

reference to this object with the newly assigned value.

6.303.3.8 virtual bool decaf::util::Date::operator== (const Date & value) const [virtual]

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.303.3.9 void decaf::util::Date::setTime (long long milliseconds)

Sets the underlying time.

Parameters

milliseconds The underlying time value in milliseconds.

6.303.3.10 std::string decaf::util::Date::toString () const

Converts this **Date** (p. 1559) object to a String of the form:

dow mon dd hh:mm:ss zzz yyyy

where:

- dow is the day of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat).
- mon is the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec).
- dd is the day of the month (01 through 31), as two decimal digits.
- hh is the hour of the day (00 through 23), as two decimal digits.

- mm is the minute within the hour (00 through 59), as two decimal digits.
- ss is the second within the minute (00 through 61, as two decimal digits.
- zzz is the time zone (and may reflect daylight saving time). Standard time zone abbreviations include those recognized by the method parse. If time zone information is not available, then zzz is empty - that is, it consists of no characters at all.
- yyyy is the year, as four decimal digits.

Returns

the String representation of the **Date** (p. 1559) object.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Date.h**

6.304 decaf::internal::DecafRuntime Class Reference

Handles APR initialization and termination.

```
#include <src/main/decaf/internal/DecafRuntime.h>
```

Inheritance diagram for decaf::internal::DecafRuntime:

Public Member Functions

- **DecafRuntime** ()
Initializes the APR Runtime for a library.
- virtual **~DecafRuntime** ()
Terminates the APR Runtime for a library.
- apr_pool_t * **getGlobalPool** () const
Grants access to the Global APR Pool instance that should be used when creating new threads.

6.304.1 Detailed Description

Handles APR initialization and termination.

6.304.2 Constructor & Destructor Documentation

6.304.2.1 decaf::internal::DecafRuntime::DecafRuntime ()

Initializes the APR Runtime for a library.

6.304.2.2 virtual decaf::internal::DecafRuntime::~~DecafRuntime () [virtual]

Terminates the APR Runtime for a library.

6.304.3 Member Function Documentation

6.304.3.1 apr_pool_t* decaf::internal::DecafRuntime::getGlobalPool () const

Grants access to the Global APR Pool instance that should be used when creating new threads.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/**DecafRuntime.h**

6.305 activemq::threads::DedicatedTaskRunner Class Reference

```
#include <src/main/activemq/threads/DedicatedTaskRunner.h>
```

Inheritance diagram for activemq::threads::DedicatedTaskRunner:

Public Member Functions

- **DedicatedTaskRunner** (Task *task)
- virtual **~DedicatedTaskRunner** ()
- virtual void **shutdown** (unsigned int timeout)

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

- virtual void **shutdown** ()

Shutdown once the task has finished and the TaskRunner's thread has exited.

- virtual void **wakeup** ()

*Signal the **TaskRunner** (p. 3496) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3494) instance will be run until its iterate method has returned false indicating it is done.*

Protected Member Functions

- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

6.305.1 Constructor & Destructor Documentation

6.305.1.1 `activemq::threads::DedicatedTaskRunner::DedicatedTaskRunner (Task * task)`

6.305.1.2 `virtual activemq::threads::DedicatedTaskRunner::~~DedicatedTaskRunner () [virtual]`

6.305.2 Member Function Documentation

6.305.2.1 `virtual void activemq::threads::DedicatedTaskRunner::run () [protected, virtual]`

Run method - called by the Thread class in the context of the thread.

Implements `decaf::lang::Runnable` (p. 3112).

6.305.2.2 `virtual void activemq::threads::DedicatedTaskRunner::shutdown () [virtual]`

Shutdown once the task has finished and the TaskRunner's thread has exited.

6.305.2.3 `virtual void activemq::threads::DedicatedTaskRunner::shutdown (unsigned int timeout) [virtual]`

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters

timeout - Time in Milliseconds to wait for the task to stop.

6.305.2.4 `virtual void activemq::threads::DedicatedTaskRunner::wakeup () [virtual]`

Signal the **TaskRunner** (p. 3496) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3494) instance will be run until its iterate method has returned false indicating it is done.

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/DedicatedTaskRunner.h`

6.306 activemq::core::policies::DefaultPrefetchPolicy Class Reference

```
#include <src/main/activemq/core/policies/DefaultPrefetchPolicy.h>
```

Inheritance diagram for `activemq::core::policies::DefaultPrefetchPolicy`:

Public Member Functions

- **DefaultPrefetchPolicy** ()
- virtual **~DefaultPrefetchPolicy** ()
- virtual void **setDurableTopicPrefetch** (int value)
Sets the amount of prefetched messages for a Durable Topic.
- virtual int **getDurableTopicPrefetch** () const
Gets the amount of messages to prefetch for a Durable Topic.
- virtual void **setQueuePrefetch** (int value)
Sets the amount of prefetched messages for a Queue.
- virtual int **getQueuePrefetch** () const
Gets the amount of messages to prefetch for a Queue.
- virtual void **setQueueBrowserPrefetch** (int value)
Sets the amount of prefetched messages for a Queue Browser.
- virtual int **getQueueBrowserPrefetch** () const
Gets the amount of messages to prefetch for a Queue Browser.
- virtual void **setTopicPrefetch** (int value)
Sets the amount of prefetched messages for a Topic.
- virtual int **getTopicPrefetch** () const
Gets the amount of messages to prefetch for a Topic.
- virtual int **getMaxPrefetchLimit** (int value) const
Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.
- virtual **PrefetchPolicy** * **clone** () const
Clone the Policy and return a new pointer to that clone.

Static Public Attributes

- static int **MAX_PREFETCH_SIZE**
- static int **DEFAULT_DURABLE_TOPIC_PREFETCH**
- static int **DEFAULT_QUEUE_PREFETCH**
- static int **DEFAULT_QUEUE_BROWSER_PREFETCH**
- static int **DEFAULT_TOPIC_PREFETCH**

6.306.1 Constructor & Destructor Documentation

6.306.1.1 `activemq::core::policies::DefaultPrefetchPolicy::DefaultPrefetchPolicy ()`

6.306.1.2 `virtual
activemq::core::policies::DefaultPrefetchPolicy::~~DefaultPrefetchPolicy () [virtual]`

6.306.2 Member Function Documentation

6.306.2.1 `virtual PrefetchPolicy* ac-
tivemq::core::policies::DefaultPrefetchPolicy::clone ()
const [virtual]`

Clone the Policy and return a new pointer to that clone.

Returns

pointer to a new **PrefetchPolicy** (p. 2783) instance that is a clone of this one.

Implements **activemq::core::PrefetchPolicy** (p. 2785).

6.306.2.2 `virtual int ac-
tivemq::core::policies::DefaultPrefetchPolicy::getDurableTopicPrefetch () const [inline, virtual]`

Gets the amount of messages to prefetch for a Durable Topic.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2785).

6.306.2.3 `virtual int ac-
tivemq::core::policies::DefaultPrefetchPolicy::getMaxPrefetchLimit (int
value) const [inline, virtual]`

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

Returns

the allowable value for a prefetch limit, either requested or the max.

Implements **activemq::core::PrefetchPolicy** (p. 2786).

6.306.2.4 `virtual int ac-
tivemq::core::policies::DefaultPrefetchPolicy::getQueueBrowserPrefetch () const [inline, virtual]`

Gets the amount of messages to prefetch for a Queue Browser.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2786).

6.306.2.5 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getQueuePrefetch ()
const [inline, virtual]`

Gets the amount of messages to prefetch for a Queue.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2786).

6.306.2.6 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getTopicPrefetch ()
const [inline, virtual]`

Gets the amount of messages to prefetch for a Topic.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2786).

6.306.2.7 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setDurableTopicPrefetch (int value) [inline, virtual]`

Sets the amount of prefetched messages for a Durable Topic.

Parameters

value The number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2786).

6.306.2.8 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueueBrowserPrefetch (int value) [inline, virtual]`

Sets the amount of prefetched messages for a Queue Browser.

Parameters

value The number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2787).

6.306.2.9 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueuePrefetch (int value)` [inline, virtual]

Sets the amount of prefetched messages for a Queue.

Parameters

value The number of messages to prefetch.

Implements `activemq::core::PrefetchPolicy` (p. 2787).

6.306.2.10 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setTopicPrefetch (int value)` [inline, virtual]

Sets the amount of prefetched messages for a Topic.

Parameters

value The number of messages to prefetch.

Implements `activemq::core::PrefetchPolicy` (p. 2787).

6.306.3 Field Documentation

6.306.3.1 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT _ - DURABLE _ TOPIC _ PREFETCH` [static]

6.306.3.2 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT _ - QUEUE _ BROWSER _ PREFETCH` [static]

6.306.3.3 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT _ - QUEUE _ PREFETCH` [static]

6.306.3.4 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT _ TOPIC _ - PREFETCH` [static]

6.306.3.5 `int activemq::core::policies::DefaultPrefetchPolicy::MAX _ - PREFETCH _ SIZE` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/policies/DefaultPrefetchPolicy.h`

6.307 activemq::core::policies::DefaultRedeliveryPolicy Class Reference

```
#include <src/main/activemq/core/policies/DefaultRedeliveryPolicy.h>
```

Inheritance diagram for `activemq::core::policies::DefaultRedeliveryPolicy`:

Public Member Functions

- **DefaultRedeliveryPolicy** ()
- virtual **~DefaultRedeliveryPolicy** ()
- virtual double **getBackOffMultiplier** () const
- virtual void **setBackOffMultiplier** (double value)
Sets the Back-Off Multiplier for Message Redelivery.
- virtual short **getCollisionAvoidancePercent** () const
- virtual void **setCollisionAvoidancePercent** (short value)
- virtual long long **getInitialRedeliveryDelay** () const
Gets the initial time that redelivery of messages is delayed.
- virtual void **setInitialRedeliveryDelay** (long long value)
Sets the initial time that redelivery will be delayed.
- virtual int **getMaximumRedeliveries** () const
Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.
- virtual void **setMaximumRedeliveries** (int value)
Sets the Maximum allowable redeliveries for a Message.
- virtual bool **isUseCollisionAvoidance** () const
- virtual void **setUseCollisionAvoidance** (bool value)
- virtual bool **isUseExponentialBackOff** () const
- virtual void **setUseExponentialBackOff** (bool value)
- virtual long long **getRedeliveryDelay** (long long previousDelay)
Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.
- virtual **RedeliveryPolicy** * **clone** () const
Create a copy of this Policy and return it.

6.307.1 Constructor & Destructor Documentation

6.307.1.1 `activemq::core::policies::DefaultRedeliveryPolicy::DefaultRedeliveryPolicy ()`

6.307.1.2 `virtual
activemq::core::policies::DefaultRedeliveryPolicy::~~DefaultRedeliveryPolicy
() [virtual]`

6.307.2 Member Function Documentation

6.307.2.1 `virtual RedeliveryPolicy* ac-
tivemq::core::policies::DefaultRedeliveryPolicy::clone () const [virtual]`

Create a copy of this Policy and return it.

Returns

pointer to a new **RedeliveryPolicy** (p. 2972) that is a copy of this one.

Implements **activemq::core::RedeliveryPolicy** (p. 2974).

6.307.2.2 `virtual double activemq::core::policies::DefaultRedeliveryPolicy::getBackOffMultiplier () const [inline, virtual]`

Returns

The value of the Back-Off Multiplier for Message Redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 2975).

6.307.2.3 `virtual short activemq::core::policies::DefaultRedeliveryPolicy::getCollisionAvoidancePercent () const [virtual]`

Returns

the currently set Collision Avoidance percentage.

Implements **activemq::core::RedeliveryPolicy** (p. 2975).

6.307.2.4 `virtual long long activemq::core::policies::DefaultRedeliveryPolicy::getInitialRedeliveryDelay () const [inline, virtual]`

Gets the initial time that redelivery of messages is delayed.

Returns

the time in milliseconds that redelivery is delayed initially.

Implements **activemq::core::RedeliveryPolicy** (p. 2975).

6.307.2.5 `virtual int activemq::core::policies::DefaultRedeliveryPolicy::getMaximumRedeliveries () const [inline, virtual]`

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

Returns

maximum allowed redeliveries for a message.

Implements **activemq::core::RedeliveryPolicy** (p. 2975).

6.307.2.6 `virtual long long activemq::core::policies::DefaultRedeliveryPolicy::getRedeliveryDelay (long long previousDelay) [virtual]`

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

Parameters

previousDelay The last delay that was used between message redeliveries.

Returns

the new delay to use before attempting another redelivery.

Implements `activemq::core::RedeliveryPolicy` (p. 2975).

6.307.2.7 `virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseCollisionAvoidance () const [inline, virtual]`

Returns

whether or not collision avoidance is enabled for this Policy.

Implements `activemq::core::RedeliveryPolicy` (p. 2976).

6.307.2.8 `virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseExponentialBackOff () const [inline, virtual]`

Returns

whether or not the exponential back off option is enabled.

Implements `activemq::core::RedeliveryPolicy` (p. 2976).

6.307.2.9 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setBackOffMultiplier (double value) [inline, virtual]`

Sets the Back-Off Multiplier for Message Redelivery.

Parameters

value The new value for the back-off multiplier.

Implements `activemq::core::RedeliveryPolicy` (p. 2976).

6.307.2.10 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setCollisionAvoidancePercent (short value) [virtual]`

Parameters

value The collision avoidance percentage setting.

Implements **activemq::core::RedeliveryPolicy** (p. 2976).

6.307.2.11 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setInitialRedeliveryDelay (long long value) [inline, virtual]`

Sets the initial time that redelivery will be delayed.

Parameters

value Time in Milliseconds to wait before starting redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 2976).

6.307.2.12 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setMaximumRedeliveries (int maximumRedeliveries) [inline, virtual]`

Sets the Maximum allowable redeliveries for a Message.

Parameters

maximumRedeliveries The maximum number of times that a message will be redelivered.

Implements **activemq::core::RedeliveryPolicy** (p. 2977).

6.307.2.13 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseCollisionAvoidance (bool value) [inline, virtual]`

Parameters

value Enable or Disable collision avoidance for this Policy.

Implements **activemq::core::RedeliveryPolicy** (p. 2977).

6.307.2.14 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseExponentialBackOff (bool value) [inline, virtual]`

Parameters

value Enable or Disable the exponential back off multiplier option.

Implements **activemq::core::RedeliveryPolicy** (p. 2977).

The documentation for this class was generated from the following file:

- src/main/activemq/core/policies/**DefaultRedeliveryPolicy.h**

6.308 decaf::internal::net::DefaultServerSocketFactory Class Reference

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

```
#include <src/main/decaf/internal/net/DefaultServerSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::DefaultServerSocketFactory:

Public Member Functions

- **DefaultServerSocketFactory** ()
- virtual **~DefaultServerSocketFactory** ()
- virtual **decaf::net::ServerSocket * createServerSocket** ()

*Create a new **ServerSocket** (p. 3136) that is unbound.
The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.*

Returns

*new **ServerSocket** (p. 3136) pointer that is owned by the caller.*

Exceptions

***IOException** if the **ServerSocket** (p. 3136) cannot be created for some reason.*
- virtual **decaf::net::ServerSocket * createServerSocket** (int port)

*Create a new **ServerSocket** (p. 3136) that is bound to the given port.
The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.*

Parameters

***port** The port to bind the **ServerSocket** (p. 3136) to.*

Returns

*new **ServerSocket** (p. 3136) pointer that is owned by the caller.*

Exceptions

***IOException** if the **ServerSocket** (p. 3136) cannot be created for some reason.*
- virtual **decaf::net::ServerSocket * createServerSocket** (int port, int backlog)

*Create a new **ServerSocket** (p. 3136) that is bound to the given port.
The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will use the specified connection backlog setting.*

Parameters

***port** The port to bind the **ServerSocket** (p. 3136) to.
backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.*

Returns

*new **ServerSocket** (p. 3136) pointer that is owned by the caller.*

Exceptions

***IOException** if the **ServerSocket** (p. 3136) cannot be created for some reason.*
- virtual **decaf::net::ServerSocket * createServerSocket** (int port, int backlog, const **decaf::net::InetAddress** *address)

Create a new **ServerSocket** (p. 3136) that is bound to the given port. The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is *NULL* than the **ServerSocket** (p. 3136) will listen on all interfaces.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.
backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.
address The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

6.308.1 Detailed Description

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

Since

1.0

6.308.2 Constructor & Destructor Documentation

6.308.2.1 decaf::internal::net::DefaultServerSocketFactory::DefaultServerSocketFactory ()

6.308.2.2 virtual decaf::internal::net::DefaultServerSocketFactory::~~DefaultServerSocketFactory () [virtual]

6.308.3 Member Function Documentation

6.308.3.1 virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket () [virtual]

Create a new **ServerSocket** (p. 3136) that is unbound.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3146).

6.308.3.2 `virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket (int port, int backlog, const decaf::net::InetAddress * address) [virtual]`

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3136) will listen on all interfaces.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

address The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3147).

6.308.3.3 `virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket (int port, int backlog) [virtual]`

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will use the specified connection backlog setting.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3147).

6.308.3.4 `virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket (int port) [virtual]`

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3147).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/DefaultServerSocketFactory.h

6.309 decaf::internal::net::DefaultSocketFactory Class Reference

SocketFactory implementation that is used to create Sockets.

```
#include <src/main/decaf/internal/net/DefaultSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::DefaultSocketFactory:

Public Member Functions

- **DefaultSocketFactory** ()
- virtual **~DefaultSocketFactory** ()
- virtual **decaf::net::Socket * createSocket** () throw (decaf::io::IOException)
*Creates an unconnected **Socket** (p. 3281) object.*
Returns
*a new **Socket** (p. 3281) object, caller must free this object when done.*
Exceptions
*IOException if the **Socket** (p. 3281) cannot be created.*
- virtual **decaf::net::Socket * createSocket** (const **decaf::net::InetAddress** *host, int port) throw (decaf::io::IOException, decaf::net::UnknownHostException)
*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*
Parameters
host The host to connect the socket to.
port The port on the remote host to connect to.
Returns
*a new **Socket** (p. 3281) object, caller must free this object when done.*
Exceptions
*IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.*

UnknownHostException (p. 3649) if the host name is not known.

- virtual **decaf::net::Socket * createSocket** (const **decaf::net::InetAddress** *host, int port, const **decaf::net::InetAddress** *ifAddress, int localPort) throw (**decaf::io::IOException**, **decaf::net::UnknownHostException**)

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

The **Socket** (p. 3281) will be bound to the specified local address and port.

Parameters

host The host to connect the socket to.

port The port on the remote host to connect to.

ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.

localPort The local port to bind the **Socket** (p. 3281) to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

- virtual **decaf::net::Socket * createSocket** (const std::string &name, int port) throw (**decaf::io::IOException**, **decaf::net::UnknownHostException**)

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host name or IP address to connect the socket to.

port The port on the remote host to connect to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

- virtual **decaf::net::Socket * createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** *ifAddress, int localPort) throw (**decaf::io::IOException**, **decaf::net::UnknownHostException**)

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host name or IP address to connect the socket to.

port The port on the remote host to connect to.

ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.

localPort The local port to bind the **Socket** (p. 3281) to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

6.309.1 Detailed Description

SocketFactory implementation that is used to create Sockets.

Since

1.0

6.309.2 Constructor & Destructor Documentation

6.309.2.1 decaf::internal::net::DefaultSocketFactory::DefaultSocketFactory ()

6.309.2.2 virtual decaf::internal::net::DefaultSocketFactory::~~DefaultSocketFactory () [virtual]

6.309.3 Member Function Documentation

6.309.3.1 virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket () throw (decaf::io::IOException) [virtual]

Creates an unconnected **Socket** (p. 3281) object.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if the **Socket** (p. 3281) cannot be created.

Reimplemented from **decaf::net::SocketFactory** (p. 3302).

6.309.3.2 virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const std::string & *name*, int *port*, const decaf::net::InetAddress * *ifAddress*, int *localPort*) throw (decaf::io::IOException, decaf::net::UnknownHostException) [virtual]

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host name or IP address to connect the socket to.

port The port on the remote host to connect to.

ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.

localPort The local port to bind the **Socket** (p. 3281) to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3303).

6.309.3.3 `virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const std::string & name, int port) throw (decaf::io::IOException, decaf::net::UnknownHostException) [virtual]`

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host name or IP address to connect the socket to.

port The port on the remote host to connect to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3303).

6.309.3.4 `virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const decaf::net::InetAddress * host, int port, const decaf::net::InetAddress * ifAddress, int localPort) throw (decaf::io::IOException, decaf::net::UnknownHostException) [virtual]`

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

The **Socket** (p. 3281) will be bound to the specified local address and port.

Parameters

host The host to connect the socket to.

port The port on the remote host to connect to.

ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.

localPort The local port to bind the **Socket** (p. 3281) to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3304).

6.309.3.5 virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const decaf::net::InetAddress * *host*, int *port*) throw (decaf::io::IOException, decaf::net::UnknownHostException) [virtual]

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host to connect the socket to.

port The port on the remote host to connect to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3302).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/DefaultSocketFactory.h

6.310 decaf::internal::net::ssl::DefaultSSLContext Class Reference

Default SSLContext manager for the Decaf library.

```
#include <src/main/decaf/internal/net/ssl/DefaultSSLContext.h>
```

Public Member Functions

- virtual ~DefaultSSLContext ()

Static Public Member Functions

- static decaf::net::ssl::SSLContext * getContext ()

Protected Member Functions

- DefaultSSLContext ()

6.310.1 Detailed Description

Default SSLContext manager for the Decaf library. If the user doesn't supply or specify the SSLContext that they wish to use then we load the Decaf library's default SSLContext using whatever SSL provider is enabled an preferred.

Since

1.0

6.310.2 Constructor & Destructor Documentation

6.310.2.1 `decaf::internal::net::ssl::DefaultSSLContext::DefaultSSLContext ()`
[protected]

6.310.2.2 `virtual decaf::internal::net::ssl::DefaultSSLContext::~~DefaultSSLContext ()` [virtual]

6.310.3 Member Function Documentation

6.310.3.1 `static decaf::net::ssl::SSLContext* decaf::internal::net::ssl::DefaultSSLContext::getContext ()` [static]

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/DefaultSSLContext.h`

6.311 decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

```
#include <src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h>
```

Inheritance diagram for `decaf::internal::net::ssl::DefaultSSLServerSocketFactory`:

Public Member Functions

- **DefaultSSLServerSocketFactory** (const std::string &errorMessage)
- virtual **~DefaultSSLServerSocketFactory** ()
- virtual **decaf::net::ServerSocket * createServerSocket** ()

*Create a new **ServerSocket** (p. 3136) that is unbound.*

*The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.*

Returns

*new **ServerSocket** (p. 3136) pointer that is owned by the caller.*

Exceptions

***IOException** if the **ServerSocket** (p. 3136) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket * createServerSocket** (int port)

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

***IOException** if the **ServerSocket** (p. 3136) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket * createServerSocket** (int port, int backlog)

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will use the specified connection backlog setting.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

***IOException** if the **ServerSocket** (p. 3136) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket * createServerSocket** (int port, int backlog, const **decaf::net::InetAddress** *address)

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is **NULL** than the **ServerSocket** (p. 3136) will listen on all interfaces.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

address The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

***IOException** if the **ServerSocket** (p. 3136) cannot be created for some reason.*

- virtual **std::vector< std::string > getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 3336)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection. Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 3336)

6.311.1 Detailed Description

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

Since

1.0

6.311.2 Constructor & Destructor Documentation

6.311.2.1 decaf::internal::net::ssl::DefaultSSLServerSocketFactory::DefaultSSLServerSocketFactory (const std::string & *errorMessage*)

6.311.2.2 virtual decaf::internal::net::ssl::DefaultSSLServerSocketFactory::~DefaultSSLServerSocketFactory () [virtual]

6.311.3 Member Function Documentation

6.311.3.1 virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket () [virtual]

Create a new **ServerSocket** (p. 3136) that is unbound.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3146).

6.311.3.2 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket (int port) [virtual]`

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3147).

6.311.3.3 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket (int port, int backlog, const decaf::net::InetAddress * address) [virtual]`

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3136) will listen on all interfaces.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

address The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3147).

6.311.3.4 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket (int port, int backlog) [virtual]`

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will use the specified connection backlog setting.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3147).

6.311.3.5 `virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLServerSocketFactory::getDefaultCipherSuites () [virtual]`

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 3336)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 3336).

6.311.3.6 `virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLServerSocketFactory::getSupportedCipherSuites () [virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 3336)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 3336).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h`

6.312 decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference

Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

```
#include <src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::ssl::DefaultSSLSocketFactory:

Public Member Functions

- **DefaultSSLSocketFactory** (const std::string &errorMessage)
- virtual ~**DefaultSSLSocketFactory** ()
- virtual **decaf::net::Socket** * **createSocket** () throw (decaf::io::IOException)
*Creates an unconnected **Socket** (p. 3281) object.*
Returns
*a new **Socket** (p. 3281) object, caller must free this object when done.*
Exceptions
*IOException if the **Socket** (p. 3281) cannot be created.*
- virtual **decaf::net::Socket** * **createSocket** (const **decaf::net::InetAddress** *host, int port) throw (decaf::io::IOException, decaf::net::UnknownHostException)
*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*
Parameters
host The host to connect the socket to.
port The port on the remote host to connect to.
Returns
*a new **Socket** (p. 3281) object, caller must free this object when done.*
Exceptions
*IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.*
UnknownHostException (p. 3649) if the host name is not known.
- virtual **decaf::net::Socket** * **createSocket** (const **decaf::net::InetAddress** *host, int port, const **decaf::net::InetAddress** *ifAddress, int localPort) throw (decaf::io::IOException, decaf::net::UnknownHostException)
*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*
*The **Socket** (p. 3281) will be bound to the specified local address and port.*
Parameters
host The host to connect the socket to.
port The port on the remote host to connect to.
ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.
localPort The local port to bind the **Socket** (p. 3281) to.
Returns
*a new **Socket** (p. 3281) object, caller must free this object when done.*

Exceptions

***IOException** if an I/O error occurs while creating the **Socket** (p. 3281) object.
UnknownHostException (p. 3649) if the host name is not known.*

- virtual **decaf::net::Socket * createSocket** (const std::string &name, int port) throw (decaf::io::IOException, decaf::net::UnknownHostException)

*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*

Parameters

***host** The host name or IP address to connect the socket to.
port The port on the remote host to connect to.*

Returns

*a new **Socket** (p. 3281) object, caller must free this object when done.*

Exceptions

***IOException** if an I/O error occurs while creating the **Socket** (p. 3281) object.
UnknownHostException (p. 3649) if the host name is not known.*

- virtual **decaf::net::Socket * createSocket** (const std::string &name, int port, const decaf::net::InetAddress *ifAddress, int localPort) throw (decaf::io::IOException, decaf::net::UnknownHostException)

*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*

Parameters

***host** The host name or IP address to connect the socket to.
port The port on the remote host to connect to.
ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.
localPort The local port to bind the **Socket** (p. 3281) to.*

Returns

*a new **Socket** (p. 3281) object, caller must free this object when done.*

Exceptions

***IOException** if an I/O error occurs while creating the **Socket** (p. 3281) object.
UnknownHostException (p. 3649) if the host name is not known.*

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

***getSupportedCipherSuites()** (p. 3347)*

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

***getDefaultCipherSuites()** (p. 3347)*

- virtual **decaf::net::Socket** * **createSocket** (**decaf::net::Socket** *socket, std::string host, int port, bool autoClose)

Returns a socket layered over an existing socket connected to the named host, at the given port. This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

socket The existing socket to layer over.
host The server host the original **Socket** (p. 3281) is connected to.
port The server port the original **Socket** (p. 3281) is connected to.
autoClose Should the layered over **Socket** (p. 3281) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 3281) instance that wraps the given **Socket** (p. 3281).

Exceptions

IOException if an I/O exception occurs while performing this operation.
UnknownHostException (p. 3649) if the host is unknown.

6.312.1 Detailed Description

Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

Since

1.0

6.312.2 Constructor & Destructor Documentation

6.312.2.1 **decaf::internal::net::ssl::DefaultSSLSocketFactory::DefaultSSLSocketFactory**
 (const std::string & *errorMessage*)

6.312.2.2 **virtual**
decaf::internal::net::ssl::DefaultSSLSocketFactory::~DefaultSSLSocketFactory
 () [virtual]

6.312.3 Member Function Documentation

6.312.3.1 **virtual decaf::net::Socket*** **decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket** () throw
 (**decaf::io::IOException**) [virtual]

Creates an unconnected **Socket** (p. 3281) object.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if the **Socket** (p. 3281) cannot be created.

Reimplemented from **decaf::net::SocketFactory** (p. 3302).

6.312.3.2 `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (decaf::net::Socket * socket, std::string host, int port, bool autoClose)` [virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

socket The existing socket to layer over.

host The server host the original **Socket** (p. 3281) is connected to.

port The server port the original **Socket** (p. 3281) is connected to.

autoClose Should the layered over **Socket** (p. 3281) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 3281) instance that wraps the given **Socket** (p. 3281).

Exceptions

IOException if an I/O exception occurs while performing this operation.

UnknownHostException (p. 3649) if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3346).

6.312.3.3 `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const decaf::net::InetAddress * host, int port) throw (decaf::io::IOException, decaf::net::UnknownHostException)` [virtual]

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host to connect the socket to.

port The port on the remote host to connect to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3302).

6.312.3.4 virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const std::string & *name*, int *port*) throw (decaf::io::IOException, decaf::net::UnknownHostException) [virtual]

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host name or IP address to connect the socket to.

port The port on the remote host to connect to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3303).

6.312.3.5 virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const std::string & *name*, int *port*, const decaf::net::InetAddress * *ifAddress*, int *localPort*) throw (decaf::io::IOException, decaf::net::UnknownHostException) [virtual]

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host name or IP address to connect the socket to.

port The port on the remote host to connect to.

ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.

localPort The local port to bind the **Socket** (p. 3281) to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3303).

6.312.3.6 `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const decaf::net::InetAddress * host, int port, const decaf::net::InetAddress * ifAddress, int localPort) throw (decaf::io::IOException, decaf::net::UnknownHostException) [virtual]`

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

The **Socket** (p. 3281) will be bound to the specified local address and port.

Parameters

host The host to connect the socket to.

port The port on the remote host to connect to.

ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.

localPort The local port to bind the **Socket** (p. 3281) to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3304).

6.312.3.7 `virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLSocketFactory::getDefaultCipherSuites () [virtual]`

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

`getSupportedCipherSuites()` (p. 3347)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3347).

6.312.3.8 `virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLSocketFactory::getSupportedCipherSuites () [virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

`getDefaultCipherSuites()` (p. 3347)

Implements `decaf::net::ssl::SSLSocketFactory` (p. 3347).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h`

6.313 activemq::transport::DefaultTransportListener Class Reference

```
#include <src/main/activemq/transport/DefaultTransportListener.h>
```

Inheritance diagram for `activemq::transport::DefaultTransportListener`:

Public Member Functions

- virtual `~DefaultTransportListener ()`
- virtual void `onCommand (const Pointer< Command > &command AMQCPP_UNUSED)`
Event handler for the receipt of a command.
- virtual void `onException (const decaf::lang::Exception &ex AMQCPP_UNUSED)`
Event handler for an exception from a command transport.
- virtual void `transportInterrupted ()`
The transport has suffered an interruption from which it hopes to recover.
- virtual void `transportResumed ()`
The transport has resumed after an interruption.

6.313.1 Constructor & Destructor Documentation

6.313.1.1 virtual
 activemq::transport::DefaultTransportListener::~~DefaultTransportListener
 () [inline, virtual]

6.313.2 Member Function Documentation

6.313.2.1 virtual void activemq::transport::DefaultTransportListener::onCommand
 (const Pointer< Command > &command *AMQCPP_UNUSED*)
 [inline, virtual]

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3629) deletes the command upon receipt.

Parameters

command the received command object.

6.313.2.2 virtual void activemq::transport::DefaultTransportListener::onException
 (const decaf::lang::Exception &ex *AMQCPP_UNUSED*) [inline,
 virtual]

Event handler for an exception from a command transport.

Parameters

ex The exception.

6.313.2.3 virtual void ac-
 tivemq::transport::DefaultTransportListener::transportInterrupted ()
 [inline, virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 3645).

6.313.2.4 virtual void ac-
 tivemq::transport::DefaultTransportListener::transportResumed ()
 [inline, virtual]

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 3645).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**DefaultTransportListener.h**

6.314 decaf::util::zip::Deflater Class Reference

This class compresses data using the *DEFLATE* algorithm (see [specification](#)).

```
#include <src/main/decaf/util/zip/Deflater.h>
```

Public Member Functions

- **Deflater** (int level, bool nowrap=false)
Creates a new compressor using the specified compression level.
- **Deflater** ()
Creates a new compressor with the default compression level.
- virtual **~Deflater** ()
- void **setInput** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)
Sets input data for compression.
- void **setInput** (const std::vector< unsigned char > &buffer, int offset, int length) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)
Sets input data for compression.
- void **setInput** (const std::vector< unsigned char > &buffer) throw (decaf::lang::exceptions::IllegalStateException)
Sets input data for compression.
- void **setDictionary** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)
Sets preset dictionary for compression.
- void **setDictionary** (const std::vector< unsigned char > &buffer, int offset, int length) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)
Sets preset dictionary for compression.
- void **setDictionary** (const std::vector< unsigned char > &buffer) throw (decaf::lang::exceptions::IllegalStateException)
Sets preset dictionary for compression.
- void **setStrategy** (int strategy) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Sets the compression strategy to the specified value.
- void **setLevel** (int level) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Sets the compression level to the specified value.

- bool **needsInput** () const
- void **finish** ()

When called, indicates that compression should end with the current contents of the input buffer.

- bool **finished** () const
- int **deflate** (unsigned char *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)

Fills specified buffer with compressed data.

- int **deflate** (std::vector< unsigned char > &buffer, int offset, int length) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)

Fills specified buffer with compressed data.

- int **deflate** (std::vector< unsigned char > &buffer) throw (decaf::lang::exceptions::IllegalStateException)

Fills specified buffer with compressed data.

- long long **getAdler** () const throw (decaf::lang::exceptions::IllegalStateException)
- long long **getBytesRead** () const throw (decaf::lang::exceptions::IllegalStateException)
- long long **getBytesWritten** () const throw (decaf::lang::exceptions::IllegalStateException)
- void **reset** () throw (decaf::lang::exceptions::IllegalStateException)

Resets deflater so that a new set of input data can be processed.

- void **end** ()

Closes the compressor and discards any unprocessed input.

Static Public Attributes

- static const int **BEST_SPEED**
Compression level for fastest compression.
- static const int **BEST_COMPRESSION**
Compression level for best compression.
- static const int **DEFAULT_COMPRESSION**
Default compression level.
- static const int **DEFLATED**
Compression method for the deflate algorithm (the only one currently supported).
- static const int **NO_COMPRESSION**
Compression level for no compression.

- static const int **FILTERED**

Compression strategy best used for data consisting mostly of small values with a somewhat random distribution.

- static const int **HUFFMAN_ONLY**

Compression strategy for Huffman coding only.

- static const int **DEFAULT_STRATEGY**

Default compression strategy.

6.314.1 Detailed Description

This class compresses data using the *DEFLATE* algorithm (see [specification](#)). Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **DeflaterOutputStream** (p. 1604) and its descendants.

The typical usage of a **Deflater** (p. 1595) instance outside this package consists of a specific call to one of its constructors before being passed to an instance of **DeflaterOutputStream** (p. 1604).

See also

DeflaterOutputStream (p. 1604)

Inflater (p. 1894)

Since

1.0

6.314.2 Constructor & Destructor Documentation

6.314.2.1 decaf::util::zip::Deflater::Deflater (int *level*, bool *nowrap* = *false*)

Creates a new compressor using the specified compression level.

If 'nowrap' is true then the ZLIB header and checksum fields will not be used in order to support the compression format used in both GZIP and PKZIP.

Parameters

level The compression level to use (0-9).

nowrap If true uses GZip compatible compression (defaults to false).

6.314.2.2 decaf::util::zip::Deflater::Deflater ()

Creates a new compressor with the default compression level.

Compressed data will be generated in ZLIB format.

6.314.2.3 virtual decaf::util::zip::Deflater::~~Deflater () [virtual]

6.314.3 Member Function Documentation

6.314.3.1 int decaf::util::zip::Deflater::deflate (unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p.1600) should be called in order to determine if more input data is required.

Parameters

buffer The Buffer to write the compressed data to.

size The size of the passed buffer.

offset The position in the Buffer to start writing at.

length The maximum number of byte of data to write.

Returns

the actual number of bytes of compressed data.

Exceptions

NullPointerException if buffer is NULL.

IndexOutOfBoundsException if the offset + length > size of the buffer.

IllegalStateException if in the end state.

6.314.3.2 int decaf::util::zip::Deflater::deflate (std::vector< unsigned char > & *buffer*, int *offset*, int *length*) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p.1600) should be called in order to determine if more input data is required.

Parameters

buffer The Buffer to write the compressed data to.

offset The position in the Buffer to start writing at.

length The maximum number of byte of data to write.

Returns

the actual number of bytes of compressed data.

Exceptions

IndexOutOfBoundsException if the offset + length > size of the buffer.

IllegalStateException if in the end state.

6.314.3.3 int decaf::util::zip::Deflater::deflate (std::vector< unsigned char > & *buffer*) throw (decaf::lang::exceptions::IllegalStateException)

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p.1600) should be called in order to determine if more input data is required.

Parameters

buffer The Buffer to write the compressed data to.

Returns

the actual number of bytes of compressed data.

Exceptions

IllegalStateException if in the end state.

6.314.3.4 void decaf::util::zip::Deflater::end ()

Closes the compressor and discards any unprocessed input.

This method should be called when the compressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Deflater** (p.1595) object is undefined.

6.314.3.5 void decaf::util::zip::Deflater::finish ()

When called, indicates that compression should end with the current contents of the input buffer.

6.314.3.6 bool decaf::util::zip::Deflater::finished () const

Returns

true if the end of the compressed data output stream has been reached.

6.314.3.7 long long decaf::util::zip::Deflater::getAdler () const throw (decaf::lang::exceptions::IllegalStateException)

Returns

the ADLER-32 value of the uncompressed data.

Exceptions

IllegalStateException if in the end state.

6.314.3.8 `long long decaf::util::zip::Deflater::getBytesRead () const throw (decaf::lang::exceptions::IllegalStateException)`

Returns

the total number of uncompressed bytes input so far.

Exceptions

IllegalStateException if in the end state.

6.314.3.9 `long long decaf::util::zip::Deflater::getBytesWritten () const throw (decaf::lang::exceptions::IllegalStateException)`

Returns

the total number of compressed bytes output so far.

Exceptions

IllegalStateException if in the end state.

6.314.3.10 `bool decaf::util::zip::Deflater::needsInput () const`

Returns

true if the input data buffer is empty and `setInput()` (p. 1602) should be called in order to provide more input

6.314.3.11 `void decaf::util::zip::Deflater::reset () throw (decaf::lang::exceptions::IllegalStateException)`

Resets deflater so that a new set of input data can be processed.

Keeps current compression level and strategy settings.

Exceptions

IllegalStateException if in the end state.

6.314.3.12 `void decaf::util::zip::Deflater::setDictionary (const std::vector< unsigned char > & buffer, int offset, int length) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)`

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with `Inflater.inflate()` (p. 1898), `Inflater.getAdler()` (p. 1896) can be called in order to get the Adler-32 value of the dictionary required for decompression.

Parameters

buffer The buffer containing the preset dictionary.
offset The position in the Buffer to start reading from.
length The number of bytes to read from the input buffer.

Exceptions

IndexOutOfBoundsException if the offset + length > size of the buffer.
IllegalStateException if in the end state.

6.314.3.13 void decaf::util::zip::Deflater::setDictionary (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p.1898), **Inflater.getAdler()** (p.1896) can be called in order to get the Adler-32 value of the dictionary required for decompression.

Parameters

buffer The buffer containing the preset dictionary.
size The size of the passed dictionary buffer.
offset The position in the Buffer to start reading from.
length The number of bytes to read from the input buffer.

Exceptions

NullPointerException if buffer is NULL.
IndexOutOfBoundsException if the offset + length > size of the buffer.
IllegalStateException if in the end state.

6.314.3.14 void decaf::util::zip::Deflater::setDictionary (const std::vector< unsigned char > & *buffer*) throw (decaf::lang::exceptions::IllegalStateException)

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p.1898), **Inflater.getAdler()** (p.1896) can be called in order to get the Adler-32 value of the dictionary required for decompression.

Parameters

buffer The buffer containing the preset dictionary.

Exceptions

IllegalStateException if in the end state.

6.314.3.15 `void decaf::util::zip::Deflater::setInput (const std::vector< unsigned char > & buffer) throw (decaf::lang::exceptions::IllegalStateException)`

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1600) returns true indicating that more input data is required.

Parameters

buffer The Buffer to read in for compression.

Exceptions

IllegalStateException if in the end state.

6.314.3.16 `void decaf::util::zip::Deflater::setInput (const std::vector< unsigned char > & buffer, int offset, int length) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)`

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1600) returns true indicating that more input data is required.

Parameters

buffer The Buffer to read in for compression.

offset The position in the Buffer to start reading from.

length The number of bytes to read from the input buffer.

Exceptions

IndexOutOfBoundsException if the offset + length > size of the buffer.

IllegalStateException if in the end state.

6.314.3.17 `void decaf::util::zip::Deflater::setInput (const unsigned char * buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)`

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1600) returns true indicating that more input data is required.

Parameters

buffer The Buffer to read in for compression.

size The size in bytes of the buffer passed.

offset The position in the Buffer to start reading from.

length The number of bytes to read from the input buffer.

Exceptions

NullPointerException if buffer is NULL.

IndexOutOfBoundsException if the offset + length > size of the buffer.

IllegalStateException if in the end state.

```
6.314.3.18 void decaf::util::zip::Deflater::setLevel ( int level )
            throw ( decaf::lang::exceptions::IllegalArgumentException,
                    decaf::lang::exceptions::IllegalStateException )
```

Sets the compression level to the specified value.

Parameters

level The new Compression level to use.

Exceptions

IllegalArgumentException if the level value is invalid.

IllegalStateException if in the end state.

```
6.314.3.19 void decaf::util::zip::Deflater::setStrategy ( int strategy )
            throw ( decaf::lang::exceptions::IllegalArgumentException,
                    decaf::lang::exceptions::IllegalStateException )
```

Sets the compression strategy to the specified value.

Parameters

strategy The new Compression strategy to use.

Exceptions

IllegalArgumentException if the strategy value is invalid.

IllegalStateException if in the end state.

6.314.4 Field Documentation

```
6.314.4.1 const int decaf::util::zip::Deflater::BEST_COMPRESSION [static]
```

Compression level for best compression.

```
6.314.4.2 const int decaf::util::zip::Deflater::BEST_SPEED [static]
```

Compression level for fastest compression.

6.314.4.3 `const int decaf::util::zip::Deflater::DEFAULT_COMPRESSION` [static]

Default compression level.

6.314.4.4 `const int decaf::util::zip::Deflater::DEFAULT_STRATEGY` [static]

Default compression strategy.

6.314.4.5 `const int decaf::util::zip::Deflater::DEFLATED` [static]

Compression method for the deflate algorithm (the only one currently supported).

6.314.4.6 `const int decaf::util::zip::Deflater::FILTERED` [static]

Compression strategy best used for data consisting mostly of small values with a somewhat random distribution.

Forces more Huffman coding and less string matching.

6.314.4.7 `const int decaf::util::zip::Deflater::HUFFMAN_ONLY` [static]

Compression strategy for Huffman coding only.

6.314.4.8 `const int decaf::util::zip::Deflater::NO_COMPRESSION` [static]

Compression level for no compression.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/Deflater.h`

6.315 `decaf::util::zip::DeflaterOutputStream` Class Reference

Provides a `FilterOutputStream` instance that compresses the data before writing it to the wrapped `OutputStream`.

```
#include <src/main/decaf/util/zip/DeflaterOutputStream.h>
```

Inheritance diagram for `decaf::util::zip::DeflaterOutputStream`:

Public Member Functions

- `DeflaterOutputStream` (`decaf::io::OutputStream *outputStream`, `bool own=false`)

*Creates a new `DeflateOutputStream` with a Default **Deflater** (p. 1595) and buffer size.*

- **DeflaterOutputStream** (decaf::io::OutputStream *outputStream, Deflater *deflater, bool own=false)

*Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1595) and a default buffer size.*

- **DeflaterOutputStream** (decaf::io::OutputStream *outputStream, Deflater *deflater, int bufferSize, bool own=false)

*Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1595) and specified buffer size.*

- virtual ~**DeflaterOutputStream** ()
- virtual void **finish** () throw (decaf::io::IOException)

Finishes writing any remaining data to the wrapped OutputStream but does not close it upon completion.

- virtual void **close** () throw (decaf::io::IOException)

*The default implementation of this method does nothing.
The close method of **FilterOutputStream** (p. 1777) calls its flush method, and then calls the close method of its underlying output stream.*

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw (decaf::io::IOException)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual void **deflate** () throw (decaf::io::IOException)

Writes a buffers worth of compressed data to the wrapped OutputStream.

Protected Attributes

- **Deflater * deflater**

*The **Deflater** (p. 1595) for this stream.*

- std::vector< unsigned char > **buf**

The Buffer to use for.

- bool **ownDeflater**
- bool **isDone**

Static Protected Attributes

- static const std::size_t **DEFAULT_BUFFER_SIZE**

6.315.1 Detailed Description

Provides a `FilterOutputStream` instance that compresses the data before writing it to the wrapped `OutputStream`.

Since

1.0

6.315.2 Constructor & Destructor Documentation

6.315.2.1 `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (decaf::io::OutputStream * outputStream, bool own = false)`

Creates a new `DeflateOutputStream` with a Default **Deflater** (p. 1595) and buffer size.

Parameters

outputStream The `OutputStream` instance to wrap.

own Should this filter take ownership of the `OutputStream` pointer (default is false).

6.315.2.2 `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (decaf::io::OutputStream * outputStream, Deflater * deflater, bool own = false)`

Creates a new `DeflateOutputStream` with a user supplied **Deflater** (p. 1595) and a default buffer size.

When the user supplied a **Deflater** (p. 1595) instance the `DeflaterOutputpotStream` does not take ownership of the **Deflater** (p. 1595) pointer, the caller is still responsible for deleting the **Deflater** (p. 1595).

Parameters

outputStream The `OutputStream` instance to wrap.

deflater The user supplied **Deflater** (p. 1595) to use for compression. (

own Should this filter take ownership of the `OutputStream` pointer (default is false).

Exceptions

NullPointerException if the **Deflater** (p. 1595) given is NULL.

6.315.2.3 `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (decaf::io::OutputStream * outputStream, Deflater * deflater, int bufferSize, bool own = false)`

Creates a new `DeflateOutputStream` with a user supplied **Deflater** (p. 1595) and specified buffer size.

When the user supplied a **Deflater** (p. 1595) instance the `DeflaterOutputpotStream` does not take ownership of the **Deflater** (p. 1595) pointer, the caller is still responsible for deleting the **Deflater** (p. 1595).

Parameters

- outputStream* The OutputStream instance to wrap.
- deflater* The user supplied **Deflater** (p.1595) to use for compression.
- bufferSize* The size of the input buffer.
- own* Should this filter take ownership of the OutputStream pointer (default is false).

Exceptions

- NullPointerException* if the **Deflater** (p.1595) given is NULL.
- IllegalArgumentException* if bufferSize is 0.

6.315.2.4 virtual decaf::util::zip::DeflaterOutputStream::~~DeflaterOutputStream () [virtual]

6.315.3 Member Function Documentation

6.315.3.1 virtual void decaf::util::zip::DeflaterOutputStream::close () throw (decaf::io::IOException) [virtual]

The default implementation of this method does nothing.

The close method of **FilterOutputStream** (p.1777) calls its flush method, and then calls the close method of its underlying output stream.

Finishes writing any remaining data to the OutputStream then closes the stream.

Reimplemented from **decaf::io::FilterOutputStream** (p.1778).

6.315.3.2 virtual void decaf::util::zip::DeflaterOutputStream::deflate () throw (decaf::io::IOException) [protected, virtual]

Writes a buffers worth of compressed data to the wrapped OutputStream.

6.315.3.3 virtual void decaf::util::zip::DeflaterOutputStream::doWriteArrayBounded (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p.1779).

6.315.3.4 virtual void decaf::util::zip::DeflaterOutputStream::doWriteByte (unsigned char *value*) throw (decaf::io::IOException) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p.1779).

6.315.3.5 `virtual void decaf::util::zip::DeflaterOutputStream::finish () throw (decaf::io::IOException)` [virtual]

Finishes writing any remaining data to the wrapped OutputStream but does not close it upon completion.

Exceptions

IOException if an I/O error occurs.

6.315.4 Field Documentation

6.315.4.1 `std::vector<unsigned char> decaf::util::zip::DeflaterOutputStream::buf` [protected]

The Buffer to use for.

6.315.4.2 `const std::size_t decaf::util::zip::DeflaterOutputStream::DEFAULT_BUFFER_SIZE` [static, protected]

6.315.4.3 `Deflater* decaf::util::zip::DeflaterOutputStream::deflater` [protected]

The **Deflater** (p.1595) for this stream.

6.315.4.4 `bool decaf::util::zip::DeflaterOutputStream::isDone` [protected]

6.315.4.5 `bool decaf::util::zip::DeflaterOutputStream::ownDeflater` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/DeflaterOutputStream.h`

6.316 decaf::util::concurrent::Delayed Class Reference

A mix-in style interface for marking objects that should be acted upon after a given delay.

```
#include <src/main/decaf/util/concurrent/Delayed.h>
```

Inheritance diagram for decaf::util::concurrent::Delayed:

Public Member Functions

- `virtual ~Delayed ()`
- `virtual long long getDelay (const TimeUnit &unit)=0`

Returns the remaining delay associated with this object, in the given time unit.

6.316.1 Detailed Description

A mix-in style interface for marking objects that should be acted upon after a given delay. An implementation of this interface must define a Comparable methods that provides an ordering consistent with its getDelay method.

6.316.2 Constructor & Destructor Documentation

6.316.2.1 virtual decaf::util::concurrent::Delayed::~~Delayed () [inline, virtual]

6.316.3 Member Function Documentation

6.316.3.1 virtual long long decaf::util::concurrent::Delayed::getDelay (const TimeUnit & *unit*) [pure virtual]

Returns the remaining delay associated with this object, in the given time unit.

Parameters

unit The time unit

Returns

the remaining delay; zero or negative values indicate that the delay has already elapsed

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/Delayed.h

6.317 cms::DeliveryMode Class Reference

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

```
#include <src/main/cms/DeliveryMode.h>
```

Public Types

- enum **DELIVERY_MODE** { **PERSISTENT** = 0, **NON_PERSISTENT** = 1 }
- Enumeration values for **Message** (p. 2375) Delivery Mode.*

Public Member Functions

- virtual ~DeliveryMode ()

6.317.1 Detailed Description

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages. When a client sends a `cms::Message` (p. 2375) it can mark the **Message** (p. 2375) as either Persistent or Non-Persistent. If the client feels that the **Message** (p. 2375) cannot be lost in transit it should mark it as Persistent, otherwise if it is allowable for a **Message** (p. 2375) to occasionally be lost it can mark it as Non-Persistent. This allows the Provider to balance make tradeoffs between balance and **Message** (p. 2375) throughput.

The **DeliveryMode** (p. 1609) covers only the transport of the **Message** (p. 2375) for sending client to its destination and doesn't apply to the receiving **Message** (p. 2375) consumer. The receiving Consumer can drop Message's based on configuration such as memory limits or **Message** (p. 2375) filtering.

A message is guaranteed to be delivered once and only once by a CMS provider if the delivery mode of the message is PERSISTENT and the configuration of the **Message** (p. 2375) consumer allows for it.

Since

1.0

6.317.2 Member Enumeration Documentation

6.317.2.1 enum cms::DeliveryMode::DELIVERY_MODE

Enumeration values for **Message** (p. 2375) Delivery Mode.

Enumerator:

PERSISTENT

NON_PERSISTENT

6.317.3 Constructor & Destructor Documentation

6.317.3.1 virtual cms::DeliveryMode::~~DeliveryMode () [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/DeliveryMode.h`

6.318 cms::Destination Class Reference

A **Destination** (p. 1610) object encapsulates a provider-specific address.

```
#include <src/main/cms/Destination.h>
```

Inheritance diagram for `cms::Destination`:

Public Types

- enum **DestinationType** { **TOPIC**, **QUEUE**, **TEMPORARY_TOPIC**, **TEMPORARY_QUEUE** }

Public Member Functions

- virtual **~Destination** ()
- virtual **DestinationType** **getDestinationType** () const =0
*Retrieve the **Destination** (p. 1610) Type for this **Destination** (p. 1610).*
- virtual **cms::Destination** * **clone** () const =0
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)=0
*Copies the contents of the given **Destination** (p. 1610) object to this one.*
- virtual const **CMSProperties** & **getCMSProperties** () const =0
Retrieve any properties that might be part of the destination that was specified.

6.318.1 Detailed Description

A **Destination** (p. 1610) object encapsulates a provider-specific address. There is no standard definition of a **Destination** (p. 1610) address, each provider can provide its own definition and there can be configuration data attached to the **Destination** (p. 1610) address.

All CMS **Destination** (p. 1610) objects support concurrent use.

Since

1.0

6.318.2 Member Enumeration Documentation

6.318.2.1 enum cms::Destination::DestinationType

Enumerator:

TOPIC

QUEUE

TEMPORARY_TOPIC

TEMPORARY_QUEUE

6.318.3 Constructor & Destructor Documentation

6.318.3.1 `virtual cms::Destination::~~Destination () [inline, virtual]`

6.318.4 Member Function Documentation

6.318.4.1 `virtual cms::Destination* cms::Destination::clone () const [pure virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implemented in `activemq::commands::ActiveMQQueue` (p. 436),
`activemq::commands::ActiveMQTempQueue` (p. 550), `ac-`
`tivemq::commands::ActiveMQTempTopic` (p. 578), and `ac-`
`tivemq::commands::ActiveMQTopic` (p. 634).

6.318.4.2 `virtual void cms::Destination::copy (const cms::Destination & source) [pure virtual]`

Copies the contents of the given **Destination** (p. 1610) object to this one.

Parameters

source The source **Destination** (p. 1610) object.

Implemented in `activemq::commands::ActiveMQQueue` (p. 436),
`activemq::commands::ActiveMQTempQueue` (p. 550), `ac-`
`tivemq::commands::ActiveMQTempTopic` (p. 578), and `ac-`
`tivemq::commands::ActiveMQTopic` (p. 634).

6.318.4.3 `virtual const CMSProperties& cms::Destination::getCMSProperties () const [pure virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

A {const} reference to a **CMSProperties** (p. 1079) object.

Implemented in `activemq::commands::ActiveMQQueue` (p. 437),
`activemq::commands::ActiveMQTempQueue` (p. 551), `ac-`
`tivemq::commands::ActiveMQTempTopic` (p. 579), and `ac-`
`tivemq::commands::ActiveMQTopic` (p. 635).

6.318.4.4 `virtual DestinationType cms::Destination::getDestinationType () const [pure virtual]`

Retrieve the **Destination** (p. 1610) Type for this **Destination** (p. 1610).

Returns

The **Destination** (p. 1610) Type

Implemented in **activemq::commands::ActiveMQQueue** (p. 438),
activemq::commands::ActiveMQTempQueue (p. 552), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 580), and **ac-**
tivemq::commands::ActiveMQTopic (p. 636).

The documentation for this class was generated from the following file:

- `src/main/cms/Destination.h`

6.319 activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

Static Public Attributes

- static const std::string **ANY_CHILD**
- static const std::string **ANY_DESCENDENT**

6.319.1 Field Documentation

6.319.1.1 const std::string
activemq::commands::ActiveMQDestination::DestinationFilter::ANY_
CHILD [static]

6.319.1.2 const std::string
activemq::commands::ActiveMQDestination::DestinationFilter::ANY_
DESCENDENT [static]

The documentation for this struct was generated from the following file:

- `src/main/activemq/commands/ActiveMQDestination.h`

6.320 activemq::commands::DestinationInfo Class Reference

```
#include <src/main/activemq/commands/DestinationInfo.h>
```

Inheritance diagram for `activemq::commands::DestinationInfo`:

Public Member Functions

- **DestinationInfo** ()
- virtual **~DestinationInfo** ()

- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DestinationInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual unsigned char **getOperationType** () const
- virtual void **setOperationType** (unsigned char operationType)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long timeout)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (**exceptions::ActiveMQException**)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_DESTINATIONINFO** = 8

Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- **Pointer**< **ActiveMQDestination** > **destination**
- unsigned char **operationType**
- long long **timeout**
- std::vector< **decaf::lang::Pointer**< **BrokerId** > > **brokerPath**

6.320.1 Constructor & Destructor Documentation

6.320.1.1 `activemq::commands::DestinationInfo::DestinationInfo ()`

6.320.1.2 `virtual activemq::commands::DestinationInfo::~~DestinationInfo ()`
[virtual]

6.320.2 Member Function Documentation

6.320.2.1 `virtual DestinationInfo* activemq::commands::DestinationInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.320.2.2 `virtual void activemq::commands::DestinationInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.320.2.3 `virtual bool activemq::commands::DestinationInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

- 6.320.2.4** `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath () const`
[virtual]
- 6.320.2.5** `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath ()` [virtual]
- 6.320.2.6** `virtual const Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId () const` [virtual]
- 6.320.2.7** `virtual Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId ()` [virtual]
- 6.320.2.8** `virtual unsigned char activemq::commands::DestinationInfo::getDataStructureType () const`
[virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSetructure** (p. 1553) type copy.

Implements **activemq::commands::DataSetructure** (p. 1557).

- 6.320.2.9 `virtual const Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination () const` [virtual]
- 6.320.2.10 `virtual Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination ()` [virtual]
- 6.320.2.11 `virtual unsigned char activemq::commands::DestinationInfo::getOperationType () const` [virtual]
- 6.320.2.12 `virtual long long activemq::commands::DestinationInfo::getTimeout () const` [virtual]
- 6.320.2.13 `virtual void activemq::commands::DestinationInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)` [virtual]
- 6.320.2.14 `virtual void activemq::commands::DestinationInfo::setConnectionId (const Pointer< ConnectionId > & connectionId)` [virtual]
- 6.320.2.15 `virtual void activemq::commands::DestinationInfo::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.320.2.16 `virtual void activemq::commands::DestinationInfo::setOperationType (unsigned char operationType)` [virtual]
- 6.320.2.17 `virtual void activemq::commands::DestinationInfo::setTimeout (long long timeout)` [virtual]
- 6.320.2.18 `virtual std::string activemq::commands::DestinationInfo::toString () const` [virtual]

Returns a string containing the information for this **DataSet** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 700).

- 6.320.2.19 `virtual Pointer<Command> activemq::commands::DestinationInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p.1112).

6.320.3 Field Documentation

- 6.320.3.1** `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::DestinationInfo::brokerPath` [protected]
- 6.320.3.2** `Pointer<ConnectionId>` `activemq::commands::DestinationInfo::connectionId`
[protected]
- 6.320.3.3** `Pointer<ActiveMQDestination>` `activemq::commands::DestinationInfo::destination`
[protected]
- 6.320.3.4** `const unsigned char` `activemq::commands::DestinationInfo::ID _ -`
`DESTINATIONINFO = 8` [static]
- 6.320.3.5** `unsigned char` `activemq::commands::DestinationInfo::operationType`
[protected]
- 6.320.3.6** `long long` `activemq::commands::DestinationInfo::timeout` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DestinationInfo.h`

6.321 `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `DestinationInfoMarshaller` (p.1618).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller`:

Public Member Functions

- `DestinationInfoMarshaller ()`
- `virtual ~DestinationInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.321.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p.1618). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.321.2 Constructor & Destructor Documentation

6.321.2.1 `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::DestinationInfoMarshaller () [inline]`

6.321.2.2 `virtual activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::~~DestinationInfoMarshaller () [inline, virtual]`

6.321.3 Member Function Documentation

6.321.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.321.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.321.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.321.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

6.321.3.5 virtual int activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 738).

6.321.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 739).

6.321.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h`

6.322 **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1622).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller**:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.322.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p.1622). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.322.2 Constructor & Destructor Documentation

6.322.2.1 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::DestinationInfoMarshaller () [inline]

6.322.2.2 virtual activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::~~DestinationInfoMarshaller () [inline, virtual]

6.322.3 Member Function Documentation

6.322.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.322.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.322.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.322.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.322.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 704).

```
6.322.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 706).

```
6.322.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h`

6.323 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1626).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller`:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.323.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1626). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.323.2 Constructor & Destructor Documentation

6.323.2.1 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::DestinationInfoMarshaller () [inline]

6.323.2.2 virtual activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::~~DestinationInfoMarshaller () [inline, virtual]

6.323.3 Member Function Documentation

6.323.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.323.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```

6.323.3.3 virtual void ac-
    tivemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseMarshal
    ( OpenWireFormat * wireFormat, commands::DataStructure *
      dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
      decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

```

6.323.3.4 virtual void ac-
    tivemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseUnmarshal
    ( OpenWireFormat * wireFormat, commands::DataStructure *
      dataStructure, decaf::io::DataInputStream * dataIn ) throw (
      decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

```

6.323.3.5 virtual int ac-
    tivemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal1
    ( OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, utils::BooleanStream * bs ) throw (
      decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 711).

```
6.323.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 712).

```
6.323.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h`

6.324 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1630).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller`:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.324.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1630). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.324.2 Constructor & Destructor Documentation

6.324.2.1 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::DestinationInfoMarshaller () [inline]

6.324.2.2 virtual
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::~~DestinationInfoMarshaller () [inline, virtual]

6.324.3 Member Function Documentation

6.324.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.324.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.324.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.324.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.324.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 718).

```
6.324.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 719).

```
6.324.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h`

6.325 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1634).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller`:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.325.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1634). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.325.2 Constructor & Destructor Documentation

6.325.2.1 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::DestinationInfoMarshaller () [inline]

6.325.2.2 virtual
activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::~~DestinationInfoMarshaller () [inline, virtual]

6.325.3 Member Function Documentation

6.325.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.325.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.325.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.325.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

6.325.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 731).

```
6.325.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 732).

```
6.325.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h`

6.326 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1638).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller`:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.326.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1638). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.326.2 Constructor & Destructor Documentation

6.326.2.1 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::DestinationInfoMarshaller () [inline]

6.326.2.2 virtual activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::~~DestinationInfoMarshaller () [inline, virtual]

6.326.3 Member Function Documentation

6.326.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.326.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.326.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.326.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

6.326.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 724).

```
6.326.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 726).

```
6.326.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h`

6.327 activemq::cmsutil::DestinationResolver Class Reference

Resolves a CMS destination name to a `Destination`.

```
#include <src/main/activemq/cmsutil/DestinationResolver.h>
```

Inheritance diagram for `activemq::cmsutil::DestinationResolver`:

Public Member Functions

- virtual `~DestinationResolver()`
- virtual void **init** (`ResourceLifecycleManager *mgr`)=0
Initializes this destination resolver for use.
- virtual void **destroy** ()=0
Destroys any allocated resources.
- virtual `cms::Destination * resolveDestinationName (cms::Session *session, const std::string &destName, bool pubSubDomain)=0` throw (`cms::CMSException`)
Resolves the given name to a destination.

6.327.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

6.327.2 Constructor & Destructor Documentation

- 6.327.2.1** virtual `activemq::cmsutil::DestinationResolver::~~DestinationResolver()` [`inline`, `virtual`]

6.327.3 Member Function Documentation

- 6.327.3.1** virtual void `activemq::cmsutil::DestinationResolver::destroy()` [`pure virtual`]

Destroys any allocated resources.

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1705).

6.327.3.2 `virtual void activemq::cmsutil::DestinationResolver::init (ResourceLifecycleManager * mgr)` [pure virtual]

Initializes this destination resolver for use.

Ensures that any previously allocated resources are first destroyed (e.g. calls `destroy()` (p. 1642)).

Parameters

mgr the resource lifecycle manager.

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1705).

6.327.3.3 `virtual cms::Destination* activemq::cmsutil::DestinationResolver::resolveDestinationName (cms::Session * session, const std::string & destName, bool pubSubDomain)` throw (`cms::CMSEException`) [pure virtual]

Resolves the given name to a destination.

If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

Parameters

session the session for which to retrieve resolve the destination.

destName the name to be resolved.

pubSubDomain If true, the name will be resolved to a Topic, otherwise a Queue.

Returns

the resolved destination

Exceptions

cms::CMSEException (p. 1074) if resolution failed.

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1705).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/DestinationResolver.h`

6.328 activemq::commands::DiscoveryEvent Class Reference

```
#include <src/main/activemq/commands/DiscoveryEvent.h>
```

Inheritance diagram for `activemq::commands::DiscoveryEvent`:

Public Member Functions

- **DiscoveryEvent** ()
- virtual **~DiscoveryEvent** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **DiscoveryEvent * cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

- virtual const std::string & **getServiceName** () const
- virtual std::string & **getServiceName** ()
- virtual void **setServiceName** (const std::string &serviceName)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)

Static Public Attributes

- static const unsigned char **ID_DISCOVERYEVENT** = 40

Protected Attributes

- std::string **serviceName**
- std::string **brokerName**

6.328.1 Constructor & Destructor Documentation

6.328.1.1 `activemq::commands::DiscoveryEvent::DiscoveryEvent ()`

6.328.1.2 `virtual activemq::commands::DiscoveryEvent::~~DiscoveryEvent ()`
[virtual]

6.328.2 Member Function Documentation

6.328.2.1 `virtual DiscoveryEvent* activemq::commands::DiscoveryEvent::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.328.2.2 `virtual void activemq::commands::DiscoveryEvent::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1555).

6.328.2.3 `virtual bool activemq::commands::DiscoveryEvent::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1556).

- 6.328.2.4** `virtual const std::string& activemq::commands::DiscoveryEvent::getBrokerName () const [virtual]`
- 6.328.2.5** `virtual std::string& activemq::commands::DiscoveryEvent::getBrokerName () [virtual]`
- 6.328.2.6** `virtual unsigned char activemq::commands::DiscoveryEvent::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.328.2.7** `virtual std::string& activemq::commands::DiscoveryEvent::getServiceName () [virtual]`
- 6.328.2.8** `virtual const std::string& activemq::commands::DiscoveryEvent::getServiceName () const [virtual]`
- 6.328.2.9** `virtual void activemq::commands::DiscoveryEvent::setBrokerName (const std::string & brokerName) [virtual]`
- 6.328.2.10** `virtual void activemq::commands::DiscoveryEvent::setServiceName (const std::string & serviceName) [virtual]`
- 6.328.2.11** `virtual std::string activemq::commands::DiscoveryEvent::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 767).

6.328.3 Field Documentation

- 6.328.3.1 `std::string activemq::commands::DiscoveryEvent::brokerName`
[protected]
- 6.328.3.2 `const unsigned char activemq::commands::DiscoveryEvent::ID_ - DISCOVERYEVENT = 40` [static]
- 6.328.3.3 `std::string activemq::commands::DiscoveryEvent::serviceName`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DiscoveryEvent.h`

6.329 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1647).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller`:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.329.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1647). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.329.2 Constructor & Destructor Documentation

6.329.2.1 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::DiscoveryEventMarshaller () [inline]

6.329.2.2 virtual activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller () [inline, virtual]

6.329.3 Member Function Documentation

6.329.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.329.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.329.3.3 virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.329.3.4 virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.329.3.5 virtual int activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

```
6.329.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

```
6.329.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller.h`

6.330 **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller** Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1651).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller**:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.330.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p.1651). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.330.2 Constructor & Destructor Documentation

6.330.2.1 `activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::DiscoveryEventMarshaller () [inline]`

6.330.2.2 `virtual activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller () [inline, virtual]`

6.330.3 Member Function Documentation

6.330.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1505).

6.330.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1511).

6.330.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1518).

```
6.330.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

```
6.330.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.330.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.330.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h`

6.331 `activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` Class Reference

Marshaling code for Open Wire Format for `DiscoveryEventMarshaller` (p. 1654).

#include <src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.331.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p.1654). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.331.2 Constructor & Destructor Documentation

6.331.2.1 `activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::DiscoveryEventM`
`() [inline]`

6.331.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::~~DiscoveryEvent`
`() [inline, virtual]`

6.331.3 Member Function Documentation

6.331.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.331.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.331.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.331.3.4 virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.331.3.5 virtual int activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.331.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.331.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h`

6.332 **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller** Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1658).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller**:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.332.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p.1658). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.332.2 Constructor & Destructor Documentation

6.332.2.1 `activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::DiscoveryEventM`
`() [inline]`

6.332.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::~~DiscoveryEvent`
`() [inline, virtual]`

6.332.3 Member Function Documentation

6.332.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.332.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.332.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.332.3.4 virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.332.3.5 virtual int activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.332.3.6 virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.332.3.7 virtual void **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h`

6.333 **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller** Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1662).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.333.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p.1662). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.333.2 Constructor & Destructor Documentation

6.333.2.1 `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::DiscoveryEventM`
`() [inline]`

6.333.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::~~DiscoveryEvent`
`() [inline, virtual]`

6.333.3 Member Function Documentation

6.333.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.333.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.333.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.333.3.4 virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.333.3.5 virtual int activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.333.3.6 virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.333.3.7 virtual void **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h`

6.334 **activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller** Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1666).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller**:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.334.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p.1666). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.334.2 Constructor & Destructor Documentation

6.334.2.1 `activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::DiscoveryEventM`
`() [inline]`

6.334.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::~~DiscoveryEvent`
`() [inline, virtual]`

6.334.3 Member Function Documentation

6.334.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.334.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.334.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.334.3.4 virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.334.3.5 virtual int activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.334.3.6 virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.334.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h`

6.335 `activemq::core::DispatchData` Class Reference

Simple POCO that contains the information necessary to route a message to a specified consumer.

```
#include <src/main/activemq/core/DispatchData.h>
```

Public Member Functions

- `DispatchData ()`
- `DispatchData (const decaf::lang::Pointer< commands::ConsumerId > &consumer, const decaf::lang::Pointer< commands::Message > &message)`
- `const decaf::lang::Pointer< commands::ConsumerId > & getConsumerId ()`
- `const decaf::lang::Pointer< commands::Message > & getMessage ()`

6.335.1 Detailed Description

Simple POCO that contains the information necessary to route a message to a specified consumer.

6.335.2 Constructor & Destructor Documentation

6.335.2.1 `activemq::core::DispatchData::DispatchData ()` [inline]

6.335.2.2 `activemq::core::DispatchData::DispatchData (const decaf::lang::Pointer< commands::ConsumerId > & consumer, const decaf::lang::Pointer< commands::Message > & message)` [inline]

6.335.3 Member Function Documentation

6.335.3.1 `const decaf::lang::Pointer< commands::ConsumerId > & activemq::core::DispatchData::getConsumerId ()` [inline]

6.335.3.2 `const decaf::lang::Pointer< commands::Message > & activemq::core::DispatchData::getMessage ()` [inline]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/DispatchData.h`

6.336 activemq::core::Dispatcher Class Reference

Interface for an object responsible for dispatching messages to consumers.

```
#include <src/main/activemq/core/Dispatcher.h>
```

Inheritance diagram for `activemq::core::Dispatcher`:

Public Member Functions

- `virtual ~Dispatcher ()`
- `virtual void dispatch (const Pointer< MessageDispatch > &message)=0`

Dispatches a message to a particular consumer.

6.336.1 Detailed Description

Interface for an object responsible for dispatching messages to consumers.

6.336.2 Constructor & Destructor Documentation

6.336.2.1 `virtual activemq::core::Dispatcher::~Dispatcher ()` [inline, virtual]

6.336.3 Member Function Documentation

6.336.3.1 `virtual void activemq::core::Dispatcher::dispatch (const Pointer< MessageDispatch > & message)` [pure virtual]

Dispatches a message to a particular consumer.

Parameters

message - the message to be dispatched.

Implemented in `activemq::core::ActiveMQConsumer` (p. 273), and `activemq::core::ActiveMQSession` (p. 476).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/Dispatcher.h`

6.337 decaf::lang::Double Class Reference

```
#include <src/main/decaf/lang/Double.h>
```

Inheritance diagram for `decaf::lang::Double`:

Public Member Functions

- **Double** (double value)
- **Double** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual **~Double** ()
- virtual int **compareTo** (const **Double** &d) const
*Compares this **Double** (p. 1672) instance with another.*
- bool **equals** (const **Double** &d) const
- virtual bool **operator==** (const **Double** &d) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Double** &d) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const double &d) const
*Compares this **Double** (p. 1672) instance with another.*
- bool **equals** (const double &d) const
- virtual bool **operator==** (const double &d) const

Compares equality between this object and the one passed.

- virtual bool **operator<** (const double &d) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.
- bool **isInfinite** () const
- bool **isNaN** () const

Static Public Member Functions

- static int **compare** (double d1, double d2)
Compares the two specified double values.
- static long long **doubleToLongBits** (double value)
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.
- static long long **doubleToRawLongBits** (double value)
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.
- static bool **isInfinite** (double value)
- static bool **isNaN** (double value)
- static double **longBitsToDouble** (long long bits)
Returns the double value corresponding to a given bit representation.
- static double **parseDouble** (const std::string value) throw (exceptions::NumberFormatException)
*Returns a new double initialized to the value represented by the specified string, as performed by the valueOf method of class **Double** (p. 1672).*

- static std::string **toHexString** (double value)
Returns a hexadecimal string representation of the double argument.
- static std::string **toString** (double value)
Returns a string representation of the double argument.
- static **Double** **valueOf** (double value)
*Returns a **Double** (p. 1672) instance representing the specified double value.*
- static **Double** **valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Double** (p. 1672) instance that wraps a primitive double which is parsed from the string value passed.*

Static Public Attributes

- static const int **SIZE** = 64
The size in bits of the primitive int type.
- static const double **MAX_VALUE**
The maximum value that the primitive type can hold.
- static const double **MIN_VALUE**
The minimum value that the primitive type can hold.
- static const double **NaN**
*Constant for the Not a **Number** (p. 2653) Value.*
- static const double **POSITIVE_INFINITY**
Constant for Positive Infinity.
- static const double **NEGATIVE_INFINITY**
Constant for Negative Infinitiy.

6.337.1 Constructor & Destructor Documentation

6.337.1.1 decaf::lang::Double::Double (double value)

Parameters

value - the primitive type to wrap

6.337.1.2 decaf::lang::Double::Double (const std::string & value) throw (exceptions::NumberFormatException)

Parameters

value - the string to convert to a primitive type to wrap

6.337.1.3 virtual decaf::lang::Double::~~Double () [inline, virtual]

6.337.2 Member Function Documentation

6.337.2.1 virtual unsigned char decaf::lang::Double::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

6.337.2.2 static int decaf::lang::Double::compare (double *d1*, double *d2*) [static]

Compares the two specified double values.

The sign of the integer value returned is the same as that of the integer that would be returned by the call: new Double(d1).compareTo(new Double(d2))

Parameters

d1 - the first double to compare

d2 - the second double to compare

Returns

the value 0 if d1 is numerically equal to d2; a value less than 0 if d1 is numerically less than d2; and a value greater than 0 if d1 is numerically greater than d2.

6.337.2.3 virtual int decaf::lang::Double::compareTo (const double & *d*) const [virtual]

Compares this **Double** (p. 1672) instance with another.

Parameters

d - the **Double** (p. 1672) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements decaf::lang::Comparable< double > (p. 1126).

6.337.2.4 virtual int decaf::lang::Double::compareTo (const Double & *d*) const [virtual]

Compares this **Double** (p. 1672) instance with another.

Parameters

d - the **Double** (p. 1672) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.337.2.5 `static long long decaf::lang::Double::doubleToLongBits (double value) [static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.

Bit 63 (the bit that is selected by the mask 0x8000000000000000L) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000ffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000L. If the argument is negative infinity, the result is 0xfff0000000000000L. If the argument is NaN, the result is 0x7ff8000000000000L.

In all cases, the result is a long integer that, when given to the `longBitsToDouble(long)` method, will produce a floating-point value the same as the argument to `doubleToLongBits` (except all NaN values are collapsed to a single "canonical" NaN value).

Parameters

value - double to be converted

Returns

the long long bits that make up the double

6.337.2.6 `static long long decaf::lang::Double::doubleToRawLongBits (double value) [static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.

Bit 63 (the bit that is selected by the mask 0x8000000000000000LL) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000ffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000LL. If the argument is negative infinity, the result is 0xfff0000000000000LL. If the argument is NaN, the result is the long integer representing the actual NaN value. Unlike the `doubleToLongBits` method, `doubleToRawLongBits` does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is a long integer that, when given to the `longBitsToDouble(long)` method, will produce a floating-point value the same as the argument to `doubleToRawLongBits`.

Parameters

value - double to be converted

Returns

the long long bits that make up the double

6.337.2.7 `virtual double decaf::lang::Double::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.337.2.8 `bool decaf::lang::Double::equals (const Double & d) const [inline]`

Parameters

d - the **Double** (p. 1672) object to compare against.

Returns

true if the two **Double** (p. 1672) Objects have the same value.

6.337.2.9 `bool decaf::lang::Double::equals (const double & d) const [inline, virtual]`

Parameters

d - the **Double** (p. 1672) object to compare against.

Returns

true if the two **Double** (p. 1672) Objects have the same value.

Implements **decaf::lang::Comparable**< **double** > (p. 1126).

6.337.2.10 `virtual float decaf::lang::Double::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.337.2.11 `virtual int decaf::lang::Double::intValue () const` [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.337.2.12 `bool decaf::lang::Double::isInfinite () const`

Returns

true if the double is equal to positive infinity.

6.337.2.13 `static bool decaf::lang::Double::isInfinite (double value)` [static]

Parameters

value - The double to check.

Returns

true if the double is equal to infinity.

6.337.2.14 `bool decaf::lang::Double::isNaN () const`

Returns

true if the double is equal to NaN.

6.337.2.15 `static bool decaf::lang::Double::isNaN (double value)` [static]

Parameters

value - The double to check.

Returns

true if the double is equal to NaN.

6.337.2.16 `static double decaf::lang::Double::longBitsToDouble (long long bits)`
[static]

Returns the double value corresponding to a given bit representation.

The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "double format" bit layout.

If the argument is 0x7ff0000000000000L, the result is positive infinity. If the argument is 0xfff0000000000000L, the result is negative infinity. If the argument is any value in the range 0x7ff0000000000001L through 0x7fffffffffffffffL or in the range 0xfff0000000000001L through 0xfffffffffffffffL, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Double.doubleToRawLongBits** (p. 1676) method.

Parameters

bits - the long long bits to convert to double

Returns

the double converted from the bits

6.337.2.17 `virtual long long decaf::lang::Double::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.337.2.18 `virtual bool decaf::lang::Double::operator< (const Double & d) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

d - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.337.2.19 `virtual bool decaf::lang::Double::operator< (const double & d) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

d - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< double >` (p. 1127).

6.337.2.20 `virtual bool decaf::lang::Double::operator==(const Double & d)
const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

d - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.337.2.21 `virtual bool decaf::lang::Double::operator==(const double & d)
const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

d - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< double >` (p.1127).

6.337.2.22 `static double decaf::lang::Double::parseDouble (const std::string value
) throw (exceptions::NumberFormatException) [static]`

Returns a new double initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Double** (p.1672).

Parameters

value - The string to parse to an double

Returns

a double parsed from the passed string

Exceptions

NumberFormatException

6.337.2.23 `virtual short decaf::lang::Double::shortValue () const [inline,
virtual]`

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

6.337.2.24 static std::string decaf::lang::Double::toHexString (double *value*) [static]

Returns a hexadecimal string representation of the double argument.

All characters mentioned below are ASCII characters.

* If the argument is NaN, the result is the string "NaN". * Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a double value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p.1954) on the exponent value. o If m is a double value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

Parameters

value - The double to convert to a string

Returns

the Hex formatted double string.

6.337.2.25 std::string decaf::lang::Double::toString () const

Returns

this **Double** (p.1672) Object as a **String** (p.3427) Representation

6.337.2.26 static std::string decaf::lang::Double::toString (double *value*) [static]

Returns a string representation of the double argument.

All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0". o If m is greater than or equal to 10⁻³ but less than 10⁷, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of m. o If m

is less than 10^{-3} or greater than or equal to 10^7 , then it is represented in so-called "computerized scientific notation." Let n be the unique integer such that $10^n \leq m < 10^{n+1}$; then let a be the mathematically exact quotient of m and 10^n so that $1 \leq a < 10$. The magnitude is then represented as the integer part of a , as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of a , followed by the letter 'E', followed by a representation of n as a decimal integer, as produced by the method **Integer.toString(int)** (p. 1954).

Parameters

value - The double to convert to a string

Returns

the formatted double string.

6.337.2.27 static Double decaf::lang::Double::valueOf (double value) [static]

Returns a **Double** (p. 1672) instance representing the specified double value.

Parameters

value - double to wrap

Returns

new **Double** (p. 1672) instance wrapping the primitive value

6.337.2.28 static Double decaf::lang::Double::valueOf (const std::string & value) throw (exceptions::NumberFormatException) [static]

Returns a **Double** (p. 1672) instance that wraps a primitive double which is parsed from the string value passed.

Parameters

value - the string to parse

Returns

a new **Double** (p. 1672) instance wrapping the double parsed from value

Exceptions

NumberFormatException on error.

6.337.3 Field Documentation

6.337.3.1 const double decaf::lang::Double::MAX_VALUE [static]

The maximum value that the primitive type can hold.

6.337.3.2 const double decaf::lang::Double::MIN_VALUE [static]

The minimum value that the primitive type can hold.

6.337.3.3 `const double decaf::lang::Double::NaN` [static]

Constant for the Not a **N**umber (p. 2653) Value.

6.337.3.4 `const double decaf::lang::Double::NEGATIVE_INFINITY` [static]

Constant for Negative Infinitiy.

6.337.3.5 `const double decaf::lang::Double::POSITIVE_INFINITY` [static]

Constant for Positive Infinity.

6.337.3.6 `const int decaf::lang::Double::SIZE = 64` [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Double.h`

6.338 decaf::internal::nio::DoubleArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/DoubleArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::DoubleArrayBuffer:

Public Member Functions

- **DoubleArrayBuffer** (int capacity, bool readOnly=false) throw (decaf::lang::exceptions::IllegalArgumentException)

*Creates a **DoubleArrayBuffer** (p. 1683) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **DoubleArrayBuffer** (double *array, int size, int offset, int length, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

*Creates a **DoubleArrayBuffer** (p. 1683) object that wraps the given array.*

- **DoubleArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int offset, int length, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

*Creates a byte buffer that wraps the passed **ByteArrayAdapter** and start at the given offset.*

- **DoubleArrayBuffer** (const **DoubleArrayBuffer** &other)

*Create a **DoubleArrayBuffer** (p. 1683) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayAdapter** and when changes are made to that data it is reflected in both.*

- virtual `~DoubleArrayBuffer ()`
- virtual `double * array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)`

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

*the array that backs this **Buffer** (p. 855).*

Exceptions

***ReadOnlyBufferException** (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.*

- virtual `int arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

***ReadOnlyBufferException** (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.*

- virtual `DoubleBuffer * asReadOnlyBuffer () const`

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only double buffer which the caller then owns.

- virtual `DoubleBuffer & compact () throw (decaf::nio::ReadOnlyBufferException)`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 860) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 859) - 1 is copied to index `n = limit()` (p. 859) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

*a reference to this **DoubleBuffer** (p. 1692).*

Exceptions

***ReadOnlyBufferException** (p. 2966) if this buffer is read-only.*

- virtual DoubleBuffer * **duplicate** ()

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*a new double **Buffer** (p. 855) which the caller owns.*

- virtual double **get** () throw (decaf::nio::BufferUnderflowException)

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the double at the current position.

Exceptions

***BufferUnderflowException** (p. 882) if there no more data to return.*

- virtual double **get** (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

Reads the value at the given index.

Parameters

***index** The index in the **Buffer** (p. 855) where the double is to be read.*

Returns

the double that is located at the given index.

Exceptions

***IndexOutOfBoundsException** if index is not smaller than the buffer's limit*

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

- virtual bool **isReadOnly** () const

- virtual DoubleBuffer & **put** (double value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

***value** The doubles value to be written.*

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual `DoubleBuffer & put (int index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)`

Writes the given doubles into this buffer at the given index.

Parameters

index *The position in the **Buffer** (p. 855) to write the data.*
value *The doubles to write.*

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException *if index greater than the buffer's limit minus the size of the type being written, or the index is negative.*
ReadOnlyBufferException (p. 2966) *if this buffer is read-only.*

- virtual `DoubleBuffer * slice () const`

*Creates a new **DoubleBuffer** (p. 1692) whose content is a shared subsequence of this buffer's content.*

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the newly create **DoubleBuffer** (p. 1692) which the caller owns.*

Protected Member Functions

- virtual void `setReadOnly (bool value)`

*Sets this **DoubleArrayBuffer** (p. 1683) as Read-Only or not Read-Only.*

6.338.1 Constructor & Destructor Documentation

6.338.1.1 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (int capacity, bool readOnly = false) throw (decaf::lang::exceptions::IllegalArgumentException)`

Creates a **DoubleArrayBuffer** (p. 1683) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

size *The size of the array, this is the limit we read and write to.*

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArgumentException if the capacity value is negative.

6.338.1.2 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (double * array, int size, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a **DoubleArrayBuffer** (p.1683) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The actual array to wrap.

size The size of the given array.

offset The position that is this buffers start position.

length The limit of how many bytes into the array this Buffer can write.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException if buffer is NULL

IndexOutOfBoundsException if offset is greater than array capacity.

6.338.1.3 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **DoubleArrayBuffer** (p.1683) will be that of the remaining capacity of the passed buffer.

Parameters

array The ByteArrayAdapter to wrap.

offset The position that is this buffers start position.

length The limit of how many bytes into the array this Buffer can write.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException if array is NULL

IndexOutOfBoundsException if offset + length is greater than array size.

6.338.1.4 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (const DoubleArrayBuffer & other)`

Create a **DoubleArrayBuffer** (p. 1683) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayAdapter` and when changes are made to that data it is reflected in both.

Parameters

other The **DoubleArrayBuffer** (p. 1683) this one is to mirror.

6.338.1.5 `virtual decaf::internal::nio::DoubleArrayBuffer::~~DoubleArrayBuffer () [virtual]`

6.338.2 Member Function Documentation

6.338.2.1 `virtual double* decaf::internal::nio::DoubleArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 855).

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implements `decaf::nio::DoubleBuffer` (p. 1696).

6.338.2.2 `virtual int decaf::internal::nio::DoubleArrayBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1696).

6.338.2.3 `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::asReadOnlyBuffer () const [virtual]`

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only double buffer which the caller then owns.

Implements **decaf::nio::DoubleBuffer** (p. 1696).

6.338.2.4 `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **DoubleBuffer** (p. 1692).

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1697).

6.338.2.5 `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::duplicate () [virtual]`

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new double **Buffer** (p. 855) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1697).

6.338.2.6 virtual double decaf::internal::nio::DoubleArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the double at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return.

Implements **decaf::nio::DoubleBuffer** (p. 1699).

6.338.2.7 virtual double decaf::internal::nio::DoubleArrayBuffer::get (int *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the double is to be read.

Returns

the double that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit

Implements **decaf::nio::DoubleBuffer** (p. 1698).

6.338.2.8 `virtual bool decaf::internal::nio::DoubleArrayBuffer::hasArray () const`
[inline, virtual]

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::DoubleBuffer** (p. 1699).

6.338.2.9 `virtual bool decaf::internal::nio::DoubleArrayBuffer::isReadOnly ()`
const [inline, virtual]

6.338.2.10 `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put`
(int *index*, double *value*) throw (de-
caf::lang::exceptions::IndexOutOfBoundsException,
decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The doubles to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or the index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1700).

6.338.2.11 `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put`
(double *value*) throw (decaf::nio::BufferOverflowException,
decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

value The doubles value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1700).

6.338.2.12 `virtual void decaf::internal::nio::DoubleArrayBuffer::setReadOnly (bool value)` [inline, protected, virtual]

Sets this **DoubleArrayBuffer** (p. 1683) as Read-Only or not Read-Only.

Parameters

value Boolean value, true if this buffer is to be read-only, false otherwise.

6.338.2.13 `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::slice () const` [virtual]

Creates a new **DoubleBuffer** (p. 1692) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **DoubleBuffer** (p. 1692) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1702).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/DoubleArrayBuffer.h`

6.339 decaf::nio::DoubleBuffer Class Reference

This class defines four categories of operations upon double buffers:

```
#include <src/main/decaf/nio/DoubleBuffer.h>
```

Inheritance diagram for **decaf::nio::DoubleBuffer**:

Public Member Functions

- virtual **~DoubleBuffer** ()
- virtual std::string **toString** () const
- virtual double * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the double array that backs this buffer (optional operation).
- virtual int **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **DoubleBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only double buffer that shares this buffer's content.
- virtual **DoubleBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **DoubleBuffer** * **duplicate** ()=0
Creates a new double buffer that shares this buffer's content.
- virtual double **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual double **get** (int index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **DoubleBuffer** & **get** (std::vector< double > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **DoubleBuffer** & **get** (double *buffer, int size, int offset, int length) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible double array.
- **DoubleBuffer** & **put** (**DoubleBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException)
This method transfers the doubles remaining in the given source buffer into this buffer.
- **DoubleBuffer** & **put** (const double *buffer, int size, int offset, int length) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)
This method transfers doubles into this buffer from the given source array.

- **DoubleBuffer** & **put** (std::vector< double > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source doubles array into this buffer.
- virtual **DoubleBuffer** & **put** (double value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual **DoubleBuffer** & **put** (int index, double value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given doubles into this buffer at the given index.
- virtual **DoubleBuffer** * **slice** () const =0
*Creates a new **DoubleBuffer** (p. 1692) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **DoubleBuffer** &value) const
- virtual bool **equals** (const **DoubleBuffer** &value) const
- virtual bool **operator==** (const **DoubleBuffer** &value) const
- virtual bool **operator<** (const **DoubleBuffer** &value) const

Static Public Member Functions

- static **DoubleBuffer** * **allocate** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)
*Allocates a new **DoubleBuffer** (p. 1692).*
- static **DoubleBuffer** * **wrap** (double *array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
*Wraps the passed buffer with a new **DoubleBuffer** (p. 1692).*
- static **DoubleBuffer** * **wrap** (std::vector< double > &buffer)
*Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1692).*

Protected Member Functions

- **DoubleBuffer** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)
*Creates a **DoubleBuffer** (p. 1692) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.339.1 Detailed Description

This class defines four categories of operations upon double buffers: o Absolute and relative get and put methods that read and write single doubles; o Relative bulk get methods that transfer contiguous sequences of doubles from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of doubles from a double array or some other double buffer into this buffer o Methods for compacting, duplicating, and slicing a double buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing double array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

Since

1.0

6.339.2 Constructor & Destructor Documentation

6.339.2.1 `decaf::nio::DoubleBuffer::DoubleBuffer (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException) [protected]`

Creates a **DoubleBuffer** (p.1692) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity The size and limit of the **Buffer** (p.855) in doubles

Exceptions

IllegalArgumentException if capacity is negative.

6.339.2.2 `virtual decaf::nio::DoubleBuffer::~~DoubleBuffer () [inline, virtual]`

6.339.3 Member Function Documentation

6.339.3.1 `static DoubleBuffer* decaf::nio::DoubleBuffer::allocate (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException) [static]`

Allocates a new **DoubleBuffer** (p.1692).

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity The size of the Double buffer in doubles.

Returns

the **DoubleBuffer** (p.1692) that was allocated, caller owns.

Exceptions

IllegalArgumentException is the capacity value is negative.

6.339.3.2 `virtual double* decaf::nio::DoubleBuffer::array () throw
(decaf::lang::exceptions::UnsupportedOperationException,
ReadOnlyBufferException) [pure virtual]`

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 855).

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in `decaf::internal::nio::DoubleArrayBuffer` (p. 1688).

6.339.3.3 `virtual int decaf::nio::DoubleBuffer::arrayOffset () throw
(decaf::lang::exceptions::UnsupportedOperationException,
ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in `decaf::internal::nio::DoubleArrayBuffer` (p. 1688).

6.339.3.4 `virtual DoubleBuffer* decaf::nio::DoubleBuffer::asReadOnlyBuffer ()
const [pure virtual]`

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow

the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only double buffer which the caller then owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1689).

6.339.3.5 virtual DoubleBuffer& decaf::nio::DoubleBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **DoubleBuffer** (p. 1692).

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1689).

6.339.3.6 virtual int decaf::nio::DoubleBuffer::compareTo (const DoubleBuffer & value) const [virtual]

6.339.3.7 virtual DoubleBuffer* decaf::nio::DoubleBuffer::duplicate () [pure virtual]

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new double **Buffer** (p. 855) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1689).

6.339.3.8 `virtual bool decaf::nio::DoubleBuffer::equals (const DoubleBuffer & value) const` [virtual]

6.339.3.9 `DoubleBuffer& decaf::nio::DoubleBuffer::get (std::vector< double > buffer) throw (BufferUnderflowException)`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than length doubles remaining in this buffer

6.339.3.10 `virtual double decaf::nio::DoubleBuffer::get (int index) const throw (lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the double is to be read.

Returns

the double that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1690).

6.339.3.11 `DoubleBuffer& decaf::nio::DoubleBuffer::get (double * buffer, int size, int offset, int length) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)`

Relative bulk get method.

This method transfers doubles from this buffer into the given destination array. If there are fewer doubles remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 860), then no bytes are transferred and a **BufferUnderflowException** (p. 882) is thrown.

Otherwise, this method copies length doubles from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters

buffer The pointer to an allocated buffer to fill.
size The size of the buffer passed.
offset The position in the buffer to start filling.
length The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than length doubles remaining in this buffer
NullPointerException if the passed buffer is null.
IndexOutOfBoundsException if the preconditions of size, offset, or length are not met.

6.339.3.12 virtual double decaf::nio::DoubleBuffer::get () throw (BufferUnderflowException) [pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the double at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1690).

6.339.3.13 virtual bool decaf::nio::DoubleBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1691).

- 6.339.3.14** `virtual bool decaf::nio::DoubleBuffer::operator< (const DoubleBuffer & value) const` [virtual]
- 6.339.3.15** `virtual bool decaf::nio::DoubleBuffer::operator== (const DoubleBuffer & value) const` [virtual]
- 6.339.3.16** `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (double value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

value The doubles value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in `decaf::internal::nio::DoubleArrayBuffer` (p. 1691).

- 6.339.3.17** `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (int index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given doubles into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The doubles to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or the index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in `decaf::internal::nio::DoubleArrayBuffer` (p. 1691).

6.339.3.18 `DoubleBuffer& decaf::nio::DoubleBuffer::put (const double
* buffer, int size, int offset, int length) throw
(BufferOverflowException, ReadOnlyBufferException,
decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::NullPointerException)`

This method transfers doubles into this buffer from the given source array.

If there are more doubles to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 860), then no doubles are transferred and a `BufferOverflowException` (p. 880) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

buffer The array from which doubles are to be read.

size The size of the buffer passed.

offset The offset within the array of the first char to be read.

length The number of doubles to be read from the given array.

Returns

a reference to this buffer.

Exceptions

`BufferOverflowException` (p. 880) if there is insufficient space in this buffer

`ReadOnlyBufferException` (p. 2966) if this buffer is read-only

`NullPointerException` if the passed buffer is null.

`IndexOutOfBoundsException` if the preconditions of `size`, `offset`, or `length` are not met.

6.339.3.19 `DoubleBuffer& decaf::nio::DoubleBuffer::put (std::vector< double > &
buffer) throw (BufferOverflowException, ReadOnlyBufferException)`

This method transfers the entire content of the given source doubles array into this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size()`.

Parameters

buffer The buffer whose contents are copied to this `DoubleBuffer` (p. 1692).

Returns

a reference to this buffer.

Exceptions

`BufferOverflowException` (p. 880) if there is insufficient space in this buffer.

`ReadOnlyBufferException` (p. 2966) if this buffer is read-only.

6.339.3.20 `DoubleBuffer& decaf::nio::DoubleBuffer::put (DoubleBuffer & src) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException)`

This method transfers the doubles remaining in the given source buffer into this buffer.

If there are more doubles remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 860), then no doubles are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies `n = src.remaining()` doubles from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

src The buffer to take doubles from an place in this one.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer for the remaining doubles in the source buffer

IllegalArgumentException if the source buffer is this buffer.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

6.339.3.21 `virtual DoubleBuffer* decaf::nio::DoubleBuffer::slice () const [pure virtual]`

Creates a new **DoubleBuffer** (p. 1692) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **DoubleBuffer** (p. 1692) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1692).

6.339.3.22 `virtual std::string decaf::nio::DoubleBuffer::toString () const [virtual]`

Returns

a `std::string` describing this object

6.339.3.23 `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (double * array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [static]`

Wraps the passed buffer with a new **DoubleBuffer** (p. 1692).

The new buffer will be backed by the given double array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array The array that will back the new buffer.

size The size of the passed in array.

offset The offset of the subarray to be used.

length The length of the subarray to be used.

Returns

a new **DoubleBuffer** (p. 1692) that is backed by `buffer`, caller owns.

Exceptions

NullPointerException if the array pointer is NULL.

IndexOutOfBoundsException if the preconditions of `size`, `offset`, or `length` are not met.

6.339.3.24 `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (std::vector< double > & buffer) [static]`

Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1692).

The new buffer will be backed by the given double array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new **DoubleBuffer** (p. 1692) that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/DoubleBuffer.h`

6.340 decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.341 activemq::cmsutil::DynamicDestinationResolver Class Reference

Resolves a CMS destination name to a `Destination`.

```
#include <src/main/activemq/cmsutil/DynamicDestinationResolver.h>
```

Inheritance diagram for `activemq::cmsutil::DynamicDestinationResolver`:

Data Structures

- class `SessionResolver`
Manages maps of names to topics and queues for a single session.

Public Member Functions

- `DynamicDestinationResolver ()`
- virtual `~DynamicDestinationResolver ()`
- virtual void `init (ResourceLifecycleManager *mgr)`
Initializes this destination resolver for use.
- virtual void `destroy ()`
Destroys any allocated resources.
- virtual `cms::Destination * resolveDestinationName (cms::Session *session, const std::string &destName, bool pubSubDomain) throw (cms::CMSException)`
Resolves the given name to a destination.

Protected Member Functions

- `DynamicDestinationResolver (const DynamicDestinationResolver &)`
- `DynamicDestinationResolver & operator= (const DynamicDestinationResolver &)`

6.341.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

6.341.2 Constructor & Destructor Documentation

- 6.341.2.1** `activemq::cmsutil::DynamicDestinationResolver::DynamicDestinationResolver (const DynamicDestinationResolver &) [inline, protected]`
- 6.341.2.2** `activemq::cmsutil::DynamicDestinationResolver::DynamicDestinationResolver ()`
- 6.341.2.3** `virtual
activemq::cmsutil::DynamicDestinationResolver::~~DynamicDestinationResolver () [virtual]`

6.341.3 Member Function Documentation

- 6.341.3.1** `virtual void activemq::cmsutil::DynamicDestinationResolver::destroy () [virtual]`

Destroys any allocated resources.

Implements `activemq::cmsutil::DestinationResolver` (p. 1642).

- 6.341.3.2** `virtual void activemq::cmsutil::DynamicDestinationResolver::init (ResourceLifecycleManager * mgr) [inline, virtual]`

Initializes this destination resolver for use.

Ensures that any previously allocated resources are first destroyed (e.g. calls `destroy()` (p. 1705)).

Parameters

mgr the resource lifecycle manager.

Implements `activemq::cmsutil::DestinationResolver` (p. 1643).

- 6.341.3.3** `DynamicDestinationResolver& activemq::cmsutil::DynamicDestinationResolver::operator= (const DynamicDestinationResolver &) [inline, protected]`
- 6.341.3.4** `virtual cms::Destination* activemq::cmsutil::DynamicDestinationResolver::resolveDestinationName (cms::Session * session, const std::string & destName, bool pubSubDomain) throw (cms::CMSEException) [virtual]`

Resolves the given name to a destination.

If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

Parameters

session the session for which to retrieve resolve the destination.

destName the name to be resolved.

pubSubDomain If true, the name will be resolved to a Topic, otherwise a Queue.

Returns

the resolved destination

Exceptions

cms::CMSException (p. 1074) if resolution failed.

Implements **activemq::cmsutil::DestinationResolver** (p. 1643).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/DynamicDestinationResolver.h`

6.342 decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference

```
#include <src/main/decaf/util/Map.h>
```

Public Member Functions

- **Entry** ()
- virtual **~Entry** ()
- virtual const K & **getKey** () const =0
- virtual const V & **getValue** () const =0
- virtual void **setValue** (const V &value)=0

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::Map< K, V, COMPARATOR >::Entry
```

6.342.1 Constructor & Destructor Documentation

6.342.1.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Entry::Entry () [inline]`

6.342.1.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::Entry::~Entry () [inline, virtual]`

6.342.2 Member Function Documentation

6.342.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const K& decaf::util::Map< K, V, COMPARATOR >::Entry::getKey () const [pure virtual]`

6.342.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::Map< K, V, COMPARATOR >::Entry::getValue () const [pure virtual]`

6.342.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::Entry::setValue (const V & value) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Map.h`

6.343 decaf::io::EOFException Class Reference

```
#include <src/main/decaf/io/EOFException.h>
```

Inheritance diagram for decaf::io::EOFException:

Public Member Functions

- **EOFException** () throw ()
Default Constructor.
- **EOFException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **EOFException** (const EOFException &ex) throw ()
Copy Constructor.

- **EOFException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **EOFException** (const std::exception *cause) throw ()
Constructor.
- **EOFException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **EOFException** * clone () const
Clones this exception.
- virtual ~**EOFException** () throw ()

6.343.1 Constructor & Destructor Documentation

6.343.1.1 decaf::io::EOFException::EOFException () throw () [inline]

Default Constructor.

6.343.1.2 decaf::io::EOFException::EOFException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.343.1.3 decaf::io::EOFException::EOFException (const EOFException & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.343.1.4 decaf::io::EOFException::EOFException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
 ... list of primitives that are formatted into the message

6.343.1.5 decaf::io::EOFException::EOFException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.343.1.6 decaf::io::EOFException::EOFException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
 ... list of primitives that are formatted into the message

6.343.1.7 virtual decaf::io::EOFException::~~EOFException () throw () [inline, virtual]

6.343.2 Member Function Documentation

6.343.2.1 virtual EOFException* decaf::io::EOFException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2005).

The documentation for this class was generated from the following file:

- src/main/decaf/io/EOFException.h

6.344 decaf::util::logging::ErrorManager Class Reference

ErrorManager (p. 1710) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1852) during Logging.

```
#include <src/main/decaf/util/logging/ErrorManager.h>
```

Public Member Functions

- **ErrorManager** ()
- virtual **~ErrorManager** ()
- virtual void **error** (const std::string &message, **decaf::lang::Exception** *ex, int code)

*The error method is called when a **Handler** (p. 1852) failure occurs.*

Static Public Attributes

- static const int **GENERIC_FAILURE**
GENERIC_FAILURE is used for failure that don't fit into one of the other categories.
- static const int **WRITE_FAILURE**
WRITE_FAILURE is used when a write to an output stream fails.
- static const int **FLUSH_FAILURE**
FLUSH_FAILURE is used when a flush to an output stream fails.
- static const int **CLOSE_FAILURE**
CLOSE_FAILURE is used when a close of an output stream fails.
- static const int **OPEN_FAILURE**
OPEN_FAILURE is used when an open of an output stream fails.
- static const int **FORMAT_FAILURE**
FORMAT_FAILURE is used when formatting fails for any reason.

6.344.1 Detailed Description

ErrorManager (p. 1710) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1852) during Logging. When processing logging output, if a **Handler** (p. 1852) encounters problems then rather than throwing an Exception back to the issuer of the logging call (who is unlikely to be interested) the **Handler** (p. 1852) should call its associated **ErrorManager** (p. 1710).

Since

1.0

6.344.2 Constructor & Destructor Documentation

6.344.2.1 `decaf::util::logging::ErrorManager::ErrorManager ()`

6.344.2.2 `virtual decaf::util::logging::ErrorManager::~~ErrorManager ()`
[virtual]

6.344.3 Member Function Documentation

6.344.3.1 `virtual void decaf::util::logging::ErrorManager::error (const std::string & message, decaf::lang::Exception * ex, int code)` [virtual]

The error method is called when a **Handler** (p. 1852) failure occurs.

This method may be overridden in subclasses. The default behavior in this base class is that the first call is reported to System.err, and subsequent calls are ignored.

Parameters

msg - a descriptive string (may be empty)

ex - an exception (may be NULL)

code - an error code defined in **ErrorManager** (p. 1710)

6.344.4 Field Documentation

6.344.4.1 `const int decaf::util::logging::ErrorManager::CLOSE_FAILURE`
[static]

CLOSE_FAILURE is used when a close of an output stream fails.

6.344.4.2 `const int decaf::util::logging::ErrorManager::FLUSH_FAILURE`
[static]

FLUSH_FAILURE is used when a flush to an output stream fails.

6.344.4.3 `const int decaf::util::logging::ErrorManager::FORMAT_FAILURE`
[static]

FORMAT_FAILURE is used when formatting fails for any reason.

6.344.4.4 `const int decaf::util::logging::ErrorManager::GENERIC_FAILURE`
[static]

GENERIC_FAILURE is used for failure that don't fit into one of the other categories.

6.344.4.5 `const int decaf::util::logging::ErrorManager::OPEN_FAILURE` [static]

OPEN_FAILURE is used when an open of an output stream fails.

6.344.4.6 `const int decaf::util::logging::ErrorManager::WRITE_FAILURE` [static]

`WRITE_FAILURE` is used when a write to an output stream fails.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/ErrorManager.h`

6.345 `decaf::lang::Exception` Class Reference

```
#include <src/main/decaf/lang/Exception.h>
```

Inheritance diagram for `decaf::lang::Exception`:

Public Member Functions

- **Exception** () throw ()
Default Constructor.
- **Exception** (const **Exception** &ex) throw ()
Copy Constructor.
- **Exception** (const std::exception *cause) throw ()
Constructor.
- **Exception** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **Exception** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual ~**Exception** () throw ()
- virtual std::string **getMessage** () const
Gets the message for this exception.
- virtual const std::exception * **getCause** () const
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- virtual void **initCause** (const std::exception *cause)
Initializes the contained cause exception with the one given.
- virtual const char * **what** () const throw ()
Implement method from std::exception.

- virtual void **setMessage** (const char *msg,...)
Sets the cause for this exception.
- virtual void **setMark** (const char *file, const int lineNumber)
Adds a file/line number to the stack trace.
- virtual **Exception** * **clone** () const
Clones this exception.
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const
Provides the stack trace for every point where this exception was caught, marked, and rethrown.
- virtual void **printStackTrace** () const
Prints the stack trace to std::err.
- virtual void **printStackTrace** (std::ostream &stream) const
Prints the stack trace to the given output stream.
- virtual std::string **getStackTraceString** () const
Gets the stack trace as one contiguous string.
- **Exception** & **operator=** (const **Exception** &ex)
Assignment operator.

Protected Member Functions

- virtual void **setStackTrace** (const std::vector< std::pair< std::string, int > > &trace)
- virtual void **buildMessage** (const char *format, va_list &args)

Protected Attributes

- std::string **message**
The cause of this exception.
- std::exception * **cause**
*The **Exception** (p. 1712) that caused this one to be thrown.*
- std::vector< std::pair< std::string, int > > **stackTrace**
The stack trace.

6.345.1 Constructor & Destructor Documentation

6.345.1.1 decaf::lang::Exception::Exception () throw ()

Default Constructor.

Referenced by decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException(),
 decaf::util::concurrent::CancellationException::CancellationException(), de-
 caf::util::concurrent::ExecutionException::ExecutionException(), de-
 caf::net::HttpRetryException::HttpRetryException(), decaf::net::MalformedURLException::MalformedURLException(),
 decaf::net::ProtocolException::ProtocolException(), decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException(),
 decaf::net::SocketTimeoutException::SocketTimeoutException(), de-
 caf::util::concurrent::TimeoutException::TimeoutException(), de-
 caf::net::UnknownHostException::UnknownHostException(), and de-
 caf::net::UnknownServiceException::UnknownServiceException().

6.345.1.2 decaf::lang::Exception::Exception (const Exception & *ex*) throw ()

Copy Constructor.

Parameters

ex The Exception (p. 1712) instance to copy.

6.345.1.3 decaf::lang::Exception::Exception (const std::exception * *cause*) throw ()

Constructor.

Parameters

***cause* Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.345.1.4 decaf::lang::Exception::Exception (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.345.1.5 decaf::lang::Exception::Exception (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- cause* The exception that was the cause for this one to be thrown.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.345.1.6 virtual decaf::lang::Exception::~Exception () throw () [virtual]

6.345.2 Member Function Documentation

6.345.2.1 virtual void decaf::lang::Exception::buildMessage (const char * *format*, va_list & *vargs*) [protected, virtual]

Referenced by decaf::lang::exceptions::NumberFormatException::NumberFormatException().

6.345.2.2 virtual Exception* decaf::lang::Exception::clone () const [virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

Copy of this **Exception** (p. 1712) object

Implements decaf::lang::Throwable (p. 3538).

Reimplemented in **activemq::exceptions::ActiveMQException** (p. 315), **activemq::exceptions::BrokerException** (p. 798), **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2689), **decaf::io::EOFException** (p. 1709), **decaf::io::InterruptedIOException** (p. 1992), **decaf::io::IOException** (p. 2005), **decaf::io::UnsupportedEncodingException** (p. 3656), **decaf::io::UTFDataFormatException** (p. 3705), **decaf::lang::exceptions::ClassCastException** (p. 1064), **decaf::lang::exceptions::IllegalArgumentException** (p. 1865), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1867), **decaf::lang::exceptions::IllegalStateException** (p. 1871), **decaf::lang::exceptions::IllegalThreadStateException** (p. 1874), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 1879), **decaf::lang::exceptions::InterruptedException** (p. 1989), **decaf::lang::exceptions::InvalidStateException** (p. 2002), **decaf::lang::exceptions::NoSuchElementException** (p. 2647), **decaf::lang::exceptions::NullPointerException** (p. 2652), **decaf::lang::exceptions::NumberFormatException** (p. 2657), **decaf::lang::exceptions::RuntimeException** (p. 3116), **decaf::lang::exceptions::UnsupportedOperationException** (p. 3659), **decaf::net::BindException** (p. 770), **decaf::net::ConnectException** (p. 1167), **decaf::net::HttpRetryException** (p. 1861), **decaf::net::MalformedURLException** (p. 2305), **decaf::net::NoRouteToHostException** (p. 2642), **decaf::net::PortUnreachableException** (p. 2783), **decaf::net::ProtocolException** (p. 2939), **decaf::net::SocketException** (p. 3300),

decaf::net::SocketTimeoutException (p. 3321), **decaf::net::UnknownHostException** (p. 3651),
decaf::net::UnknownServiceException (p. 3654), **decaf::net::URISyntaxException** (p. 3689),
decaf::nio::BufferOverflowException (p. 882), **decaf::nio::BufferUnderflowException** (p. 884), **decaf::nio::InvalidMarkException** (p. 1999),
decaf::nio::ReadOnlyBufferException (p. 2968), **decaf::security::cert::CertificateEncodingException** (p. 1011),
decaf::security::cert::CertificateException (p. 1013), **decaf::security::cert::CertificateExpiredException** (p. 1015),
decaf::security::cert::CertificateNotYetValidException (p. 1017),
decaf::security::cert::CertificateParsingException (p. 1019),
decaf::security::GeneralSecurityException (p. 1847), **decaf::security::InvalidKeyException** (p. 1996),
decaf::security::KeyException (p. 2153), **decaf::security::KeyManagementException** (p. 2155),
decaf::security::NoSuchAlgorithmException (p. 2645), **decaf::security::NoSuchProviderException** (p. 2650),
decaf::security::SignatureException (p. 3278), **decaf::util::concurrent::BrokenBarrierException** (p. 792),
decaf::util::concurrent::CancellationException (p. 1006), **decaf::util::concurrent::ExecutionException** (p. 1749),
decaf::util::concurrent::RejectedExecutionException (p. 2986), **decaf::util::concurrent::TimeoutException** (p. 3543),
decaf::util::zip::DataFormatException (p. 1452), and **decaf::util::zip::ZipException** (p. 3798).

6.345.2.3 `virtual const std::exception* decaf::lang::Exception::getCause () const`
`[inline, virtual]`

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implements **decaf::lang::Throwable** (p. 3539).

6.345.2.4 `virtual std::string decaf::lang::Exception::getMessage () const`
`[inline, virtual]`

Gets the message for this exception.

Returns

Text formatted error message

Implements **decaf::lang::Throwable** (p. 3539).

6.345.2.5 `virtual std::vector< std::pair< std::string, int> > decaf::lang::Exception::getStackTrace () const` `[virtual]`

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

The first item in the returned vector is the first point where the mark was set (e.g. where the exception was created).

Returns

the stack trace.

Implements **decaf::lang::Throwable** (p. 3539).

6.345.2.6 virtual std::string decaf::lang::Exception::getStackTraceString () const [virtual]

Gets the stack trace as one contiguous string.

Returns

string with formatted stack trace data

Implements **decaf::lang::Throwable** (p. 3539).

6.345.2.7 virtual void decaf::lang::Exception::initCause (const std::exception * *cause*) [virtual]

Initializes the contained cause exception with the one given.

A copy is made to avoid ownership issues.

Parameters

cause The exception that was the cause of this one.

Implements **decaf::lang::Throwable** (p. 3540).

6.345.2.8 Exception& decaf::lang::Exception::operator= (const Exception & *ex*)

Assignment operator.

Parameters

ex const reference to another **Exception** (p. 1712)

6.345.2.9 virtual void decaf::lang::Exception::printStackTrace () const [virtual]

Prints the stack trace to std::err.

Implements **decaf::lang::Throwable** (p. 3540).

6.345.2.10 virtual void decaf::lang::Exception::printStackTrace (std::ostream & *stream*) const [virtual]

Prints the stack trace to the given output stream.

Parameters

stream the target output stream.

Implements **decaf::lang::Throwable** (p. 3540).

6.345.2.11 `virtual void decaf::lang::Exception::setMark (const char * file, const int lineNumber)` [virtual]

Adds a file/line number to the stack trace.

Parameters

file The name of the file calling this method (use `__FILE__`).

lineNumber The line number in the calling file (use `__LINE__`).

Implements **decaf::lang::Throwable** (p. 3540).

Referenced by `activemq::exceptions::BrokerException::BrokerException()`, and `decaf::lang::exceptions::NumberFormatException::NumberFormatException()`.

6.345.2.12 `virtual void decaf::lang::Exception::setMessage (const char * msg, ...)` [virtual]

Sets the cause for this exception.

Parameters

msg the format string for the msg.

... params to format into the string

Referenced by `activemq::exceptions::BrokerException::BrokerException()`.

6.345.2.13 `virtual void decaf::lang::Exception::setStackTrace (const std::vector< std::pair< std::string, int > > & trace)` [protected, virtual]

6.345.2.14 `virtual const char* decaf::lang::Exception::what () const throw ()` [inline, virtual]

Implement method from `std::exception`.

Returns

the const char* of `getMessage()` (p. 1716).

6.345.3 Field Documentation

6.345.3.1 `std::exception* decaf::lang::Exception::cause` [protected]

The **Exception** (p. 1712) that caused this one to be thrown.

6.345.3.2 `std::string decaf::lang::Exception::message` [protected]

The cause of this exception.

6.345.3.3 `std::vector< std::pair< std::string, int> > decaf::lang::Exception::stackTrace` [protected]

The stack trace.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Exception.h`

6.346 cms::ExceptionListener Class Reference

If a CMS provider detects a serious problem, it notifies the client application through an `ExceptionListener` (p. 1719) that is registered with the `Connection` (p. 1168).

```
#include <src/main/cms/ExceptionListener.h>
```

Public Member Functions

- virtual `~ExceptionListener()`
- virtual void `onException` (const `cms::CMSException` &ex)=0
Called when an exception occurs.

6.346.1 Detailed Description

If a CMS provider detects a serious problem, it notifies the client application through an `ExceptionListener` (p. 1719) that is registered with the `Connection` (p. 1168). An exception listener allows a client to be notified of a problem asynchronously. Some connections only consume messages via the asynchronous event mechanism so they would have no other way to learn that their connection has failed.

Since

1.0

6.346.2 Constructor & Destructor Documentation**6.346.2.1** `virtual cms::ExceptionListener::~ExceptionListener ()` [inline, virtual]**6.346.3 Member Function Documentation****6.346.3.1** `virtual void cms::ExceptionListener::onException (const cms::CMSException & ex)` [pure virtual]

Called when an exception occurs.

Once notified of an exception the caller should no longer use the resource that generated the exception.

Parameters

- ex* Exception Object that occurred.

The documentation for this class was generated from the following file:

- src/main/cms/**ExceptionListener.h**

6.347 activemq::commands::ExceptionResponse Class Reference

```
#include <src/main/activemq/commands/ExceptionResponse.h>
```

Inheritance diagram for activemq::commands::ExceptionResponse:

Public Member Functions

- **ExceptionResponse** ()
- virtual **~ExceptionResponse** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ExceptionResponse * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)

Static Public Attributes

- static const unsigned char **ID_EXCEPTIONRESPONSE** = 31

Protected Attributes

- `Pointer< BrokerError > exception`

6.347.1 Constructor & Destructor Documentation

6.347.1.1 `activemq::commands::ExceptionResponse::ExceptionResponse ()`

6.347.1.2 `virtual activemq::commands::ExceptionResponse::~~ExceptionResponse ()` [virtual]

6.347.2 Member Function Documentation

6.347.2.1 `virtual ExceptionResponse* activemq::commands::ExceptionResponse::cloneDataStructure ()` const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p.3077).

6.347.2.2 `virtual void activemq::commands::ExceptionResponse::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::Response` (p.3078).

6.347.2.3 `virtual bool activemq::commands::ExceptionResponse::equals (const DataStructure * value)` const [virtual]

Compares the `DataStructure` (p.1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::Response` (p.3078).

6.347.2.4 `virtual unsigned char activemq::commands::ExceptionResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Reimplemented from **activemq::commands::Response** (p. 3078).

6.347.2.5 `virtual Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException () [virtual]`

6.347.2.6 `virtual const Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException () const [virtual]`

6.347.2.7 `virtual void activemq::commands::ExceptionResponse::setException (const Pointer< BrokerError > & exception) [virtual]`

6.347.2.8 `virtual std::string activemq::commands::ExceptionResponse::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3079).

6.347.3 Field Documentation

6.347.3.1 `Pointer<BrokerError> activemq::commands::ExceptionResponse::exception [protected]`

6.347.3.2 `const unsigned char activemq::commands::ExceptionResponse::ID_ - EXCEPTIONRESPONSE = 31 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ExceptionResponse.h`

6.348 **activemq::wireformat::openwire::marshal::v6::ExceptionResponse** Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1722).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ExceptionResponseMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.348.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p.1722). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.348.2 Constructor & Destructor Documentation

6.348.2.1 `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::ExceptionRe`
`() [inline]`

6.348.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::~ExceptionR`
`() [inline, virtual]`

6.348.3 Member Function Documentation

6.348.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller`
 (p. 3108).

6.348.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller`
 (p. 3109).

6.348.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109).

6.348.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109).

6.348.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3110).

6.348.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3110).

6.348.3.7 `virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3111).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ExceptionResponseMarshaller.h`

6.349 `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `ExceptionResponseMarshaller` (p. 1726).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.349.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p.1726). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.349.2 Constructor & Destructor Documentation

6.349.2.1 `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::ExceptionRe`
`() [inline]`

6.349.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::~ExceptionR`
`() [inline, virtual]`

6.349.3 Member Function Documentation

6.349.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3090).

6.349.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3091).

6.349.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3091).

6.349.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3091).

6.349.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3092).

6.349.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3092).

6.349.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3093).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h`

6.350 `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `ExceptionResponseMarshaller` (p. 1730).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.350.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p.1730). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.350.2 Constructor & Destructor Documentation

6.350.2.1 `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::ExceptionRe`
`() [inline]`

6.350.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::~ExceptionR`
`() [inline, virtual]`

6.350.3 Member Function Documentation

6.350.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller`
 (p. 3099).

6.350.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller`
 (p. 3100).

6.350.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3100).

6.350.3.4 **virtual void ac-**
tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseUnmarsh
(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3100).

6.350.3.5 **virtual int ac-**
tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, utils::BooleanStream * *bs*) throw (**
decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3101).

6.350.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3101).

6.350.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3102).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h`

6.351 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for `ExceptionResponseMarshaller` (p. 1734).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.351.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p.1734). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.351.2 Constructor & Destructor Documentation

6.351.2.1 `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::ExceptionRe`
`() [inline]`

6.351.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::~ExceptionR`
`() [inline, virtual]`

6.351.3 Member Function Documentation

6.351.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3095).

6.351.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3095).

6.351.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3095).

6.351.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3096).

6.351.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3096).

6.351.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3097).

6.351.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3097).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h`

6.352 `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `ExceptionResponseMarshaller` (p. 1738).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.352.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p.1738). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.352.2 Constructor & Destructor Documentation

6.352.2.1 `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::ExceptionRe`
`() [inline]`

6.352.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::~ExceptionR`
`() [inline, virtual]`

6.352.3 Member Function Documentation

6.352.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3086).

6.352.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3086).

6.352.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3086).

6.352.3.4 **virtual void ac-**
tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseUnmarsh
(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3087).

6.352.3.5 **virtual int ac-**
tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, utils::BooleanStream * *bs*) throw (**
decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3087).

6.352.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3088).

6.352.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3088).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h`

6.353 `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `ExceptionResponseMarshaller` (p. 1742).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.353.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p.1742). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.353.2 Constructor & Destructor Documentation

6.353.2.1 `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::ExceptionRe`
`() [inline]`

6.353.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::~ExceptionR`
`() [inline, virtual]`

6.353.3 Member Function Documentation

6.353.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller`
 (p. 3104).

6.353.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller`
 (p. 3104).

6.353.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**
(p. 3104).

6.353.3.4 **virtual void ac-**
tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseUnmarsha
(OpenWireFormat * *wireFormat*, commands::DataStructure *
***dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (**
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**
(p. 3105).

6.353.3.5 **virtual int ac-**
tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal1
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, utils::BooleanStream * *bs*) throw (**
decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**
(p. 3105).

```

6.353.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal2
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3106).

```

6.353.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightUnmarsh
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataInputStream * dataIn,
    utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3106).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h`

6.354 decaf::util::concurrent::ExecutionException Class Reference

```
#include <src/main/decaf/util/concurrent/ExecutionException.h>
```

Inheritance diagram for decaf::util::concurrent::ExecutionException:

Public Member Functions

- **ExecutionException** () throw ()
Default Constructor.
- **ExecutionException** (const **decaf::lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **ExecutionException** (const **ExecutionException** &ex) throw ()
Copy Constructor.
- **ExecutionException** (const std::exception *cause) throw ()
Constructor.
- **ExecutionException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ExecutionException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ExecutionException** * clone () const
Clones this exception.
- virtual ~**ExecutionException** () throw ()

6.354.1 Constructor & Destructor Documentation

6.354.1.1 decaf::util::concurrent::ExecutionException::ExecutionException ()
throw () [inline]

Default Constructor.

6.354.1.2 decaf::util::concurrent::ExecutionException::ExecutionException (const
decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex - An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.354.1.3 `decaf::util::concurrent::ExecutionException::ExecutionException (const ExecutionException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex - The Exception to copy in this new instance.

References `decaf::lang::Exception::Exception()`.

6.354.1.4 `decaf::util::concurrent::ExecutionException::ExecutionException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.354.1.5 `decaf::util::concurrent::ExecutionException::ExecutionException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

msg - The message to report

... - The list of primitives that are formatted into the message

6.354.1.6 `decaf::util::concurrent::ExecutionException::ExecutionException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

cause - The exception that was the cause for this one to be thrown.

msg - The message to report

... - list of primitives that are formatted into the message

6.354.1.7 virtual decaf::util::concurrent::ExecutionException::~~ExecutionException () throw () [inline, virtual]

6.354.2 Member Function Documentation

6.354.2.1 virtual ExecutionException* decaf::util::concurrent::ExecutionException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ExecutionException.h**

6.355 decaf::util::concurrent::Executor Class Reference

An object that executes submitted **decaf.lang Runnable** (p. 3111) tasks.

```
#include <src/main/decaf/util/concurrent/Executor.h>
```

Inheritance diagram for decaf::util::concurrent::Executor:

Public Member Functions

- virtual ~**Executor** ()
- virtual void **execute** (Runnable *command)=0 throw (decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException)

Executes the given command at some time in the future.

6.355.1 Detailed Description

An object that executes submitted **decaf.lang Runnable** (p. 3111) tasks. This interface provides a way of decoupling task submission from the mechanics of how each task will be run, including details of thread use, scheduling, etc. An **Executor** (p. 1749) is normally used instead of explicitly creating threads. For example, rather than invoking `new Thread(new RunnableTask()).start()` for each of a set of tasks, you might use:

```
Executor (p. 1749) executor = anExecutor;
```

```

executor->execute( new RunnableTask1() );
executor->execute( new RunnableTask2() );
...

```

However, the `Executor` (p.1749) interface does not strictly require that execution be asynchronous. In the simplest case, an executor can run the submitted task immediately in the caller's thread:

```

class DirectExecutor : public Executor (p.1749) {
public:

    void execute( Runnable* r ) (p.1750) {
        r->run();
    }

}

```

More typically, tasks are executed in some thread other than the caller's thread. The executor below spawns a new thread for each task.

```

class ThreadPerTaskExecutor : public Executor (p.1749) {
public:
    std::vector<Thread*gt; threads;

    void execute( Runnable* r ) (p.1750) {
        threads.push_back( new Thread( r ) );
        threads.rbegin()->start();
    }

}

```

The `Executor` (p.1749) implementations provided in this package implement **decaf.util.concurrent.ExecutorService** (p.1751), which is a more extensive interface. The **decaf.util.concurrent.ThreadPoolExecutor** (p.??) class provides an extensible thread pool implementation. The **decaf.util.concurrent.Executor** (p.??) class provides convenient factory methods for these Executors.

Since

1.0

6.355.2 Constructor & Destructor Documentation

6.355.2.1 `virtual decaf::util::concurrent::Executor::~~Executor ()` [inline, virtual]

6.355.3 Member Function Documentation

6.355.3.1 `virtual void decaf::util::concurrent::Executor::execute (Runnable * command) throw (decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException)` [pure virtual]

Executes the given command at some time in the future.

The command may execute in a new thread, in a pooled thread, or in the calling thread, at the discretion of the **Executor** (p. 1749) implementation.

Parameters

command the runnable task

Exceptions

RejectedExecutionException (p. 2984) if this task cannot be accepted for execution.

NullPointerException if command is null

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Executor.h**

6.356 decaf::util::concurrent::ExecutorService Class Reference

An **Executor** (p. 1749) that provides methods to manage termination and methods that can produce a **Future** (p. 1840) for tracking progress of one or more asynchronous tasks.

```
#include <src/main/decaf/util/concurrent/ExecutorService.h>
```

Inheritance diagram for decaf::util::concurrent::ExecutorService:

Public Member Functions

- virtual **~ExecutorService** ()
- bool **awaitTermination** (long long timeout, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::InterruptedException)
Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.

6.356.1 Detailed Description

An **Executor** (p. 1749) that provides methods to manage termination and methods that can produce a **Future** (p. 1840) for tracking progress of one or more asynchronous tasks. An **ExecutorService** (p. 1751) can be shut down, which will cause it to reject new tasks. Two different methods are provided for shutting down an **ExecutorService** (p. 1751). The shutdown() method will allow previously submitted tasks to execute before terminating, while the shutdownNow() method prevents waiting tasks from starting and attempts to stop currently executing tasks. Upon termination, an executor has no tasks actively executing, no tasks awaiting execution, and no new tasks can be submitted. An unused **ExecutorService** (p. 1751) should be shut down to allow reclamation of its resources.

Method submit extends base method **Executor.execute** (p. 1750)(decaf.lang.**Runnable** (p. 3111)) by creating and returning a **Future** (p. 1840) that can be used to cancel execution

and/or wait for completion. Methods `invokeAny` and `invokeAll` perform the most commonly useful forms of bulk execution, executing a collection of tasks and then waiting for at least one, or all, to complete. (Class `ExecutorCompletionService` can be used to write customized variants of these methods.)

The `Executors` class provides factory methods for the executor services provided in this package.

Since

1.0

6.356.2 Constructor & Destructor Documentation

6.356.2.1 `virtual decaf::util::concurrent::ExecutorService::~~ExecutorService ()`
[inline, virtual]

6.356.3 Member Function Documentation

6.356.3.1 `bool decaf::util::concurrent::ExecutorService::awaitTermination`
(`long long timeout`, `const TimeUnit & unit`) `throw (`
`decaf::lang::exceptions::InterruptedException)` [pure virtual]

Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.

Parameters

timeout The amount of time to wait before timing out the Wait operation.

unit The Units that comprise the timeout value.

Returns

true if the executor terminated before the given timeout value elapsed.

Exceptions

InterruptedException - if interrupted while waiting.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ExecutorService.h`

6.357 activemq::transport::failover::FailoverTransport Class Reference

```
#include <src/main/activemq/transport/failover/FailoverTransport.h>
```

Inheritance diagram for `activemq::transport::failover::FailoverTransport`:

Public Member Functions

- **FailoverTransport** ()
- virtual **~FailoverTransport** ()
- void **reconnect** ()
*Indicates that the **Transport** (p. 3629) needs to reconnect to another URI in its list.*
- void **add** (const std::string &uri)
Adds a New URI to the List of URIs this transport can Connect to.
- virtual void **addURI** (const List< URI > &uris)
*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3629) is a composite of.*
- virtual void **removeURI** (const List< URI > &uris)
*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3629) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3629) should result in that **Transport** (p. 3629) being disposed of.*
- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **stop** () throw (decaf::io::IOException)
*Stop the **Transport** (p. 3629).*
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.
- virtual void **oneway** (const Pointer< Command > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual Pointer< Response > **request** (const Pointer< Command > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual Pointer< Response > **request** (const Pointer< Command > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual void **setWireFormat** (const Pointer< wireformat::WireFormat > &wireFormat AMQCPP_UNUSED)
Sets the WireFormat instance to use.
- virtual void **setTransportListener** (TransportListener *listener)
Sets the observer of asynchronous events from this transport.
- virtual TransportListener * **getTransportListener** () const
Gets the observer of asynchronous exceptions from this transport.

- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
*Is the **Transport** (p. 3629) Connected to its Broker.*
- virtual bool **isClosed** () const
*Has the **Transport** (p. 3629) been shutdown and no longer usable.*
- bool **isInitialized** () const
*Returns true if the **Transport** (p. 3629) has been initialized by a BrokerInfo command.*
- void **setInitialized** (bool value)
*Sets the initialized state of this **Transport** (p. 3629) to true.*
- virtual **Transport** * **narrow** (const std::type_info &typeId)
*Narrows down a Chain of Transports to a specific **Transport** (p. 3629) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual std::string **getRemoteAddress** () const
- virtual bool **isPending** () const
- virtual bool **iterate** ()
*Performs the actual Reconnect operation for the **FailoverTransport** (p. 1752), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.*
- virtual void **reconnect** (const decaf::net::URI &uri) throw (decaf::io::IOException)
reconnect to another location
- long long **getTimeout** () const
- void **setTimeout** (long long value)
- long long **getInitialReconnectDelay** () const
- void **setInitialReconnectDelay** (long long value)
- long long **getMaxReconnectDelay** () const
- void **setMaxReconnectDelay** (long long value)
- long long **getBackOffMultiplier** () const
- void **setBackOffMultiplier** (long long value)
- bool **isUseExponentialBackOff** () const
- void **setUseExponentialBackOff** (bool value)
- bool **isRandomize** () const
- void **setRandomize** (bool value)
- int **getMaxReconnectAttempts** () const
- void **setMaxReconnectAttempts** (int value)
- int **getStartupMaxReconnectAttempts** () const
- void **setStartupMaxReconnectAttempts** (int value)
- long long **getReconnectDelay** () const
- void **setReconnectDelay** (long long value)
- bool **isBackup** () const
- void **setBackup** (bool value)
- int **getBackupPoolSize** () const

- void **setBackupPoolSize** (int value)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool value)
- bool **isTrackTransactionProducers** () const
- void **setTrackTransactionProducers** (bool value)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int value)
- void **setConnectionInterruptProcessingComplete** (const **Pointer**< **commands::ConnectionId** > &connectionId)

Protected Member Functions

- void **restoreTransport** (const **Pointer**< **Transport** > &transport) throw (decaf::io::IOException)
*Given a **Transport** (p. 3629) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.*
- void **handleTransportFailure** (const decaf::lang::Exception &error) throw (decaf::lang::Exception)
*Called when this class' **TransportListener** (p. 3643) is notified of a Failure.*

Friends

- class **FailoverTransportListener**

6.357.1 Constructor & Destructor Documentation

6.357.1.1 **activemq::transport::failover::FailoverTransport::FailoverTransport** ()

6.357.1.2 **virtual**
activemq::transport::failover::FailoverTransport::~~FailoverTransport ()
 [virtual]

6.357.2 Member Function Documentation

6.357.2.1 **void activemq::transport::failover::FailoverTransport::add** (const std::string & *uri*)

Adds a New URI to the List of URIs this transport can Connect to.

Parameters

uri A String version of a URI to add to the URIs to failover to.

6.357.2.2 **virtual void activemq::transport::failover::FailoverTransport::addURI** (const List< URI > & *uris*) [virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3629) is a composite of.

Parameters

uris The new URIs to add to the set this composite maintains.

6.357.2.3 `virtual void activemq::transport::failover::FailoverTransport::close ()
throw (decaf::io::IOException) [virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if errors occur.

6.357.2.4 `long long ac-
tivemq::transport::failover::FailoverTransport::getBackOffMultiplier ()
const [inline]`

6.357.2.5 `int activemq::transport::failover::FailoverTransport::getBackupPoolSize ()
const [inline]`

6.357.2.6 `long long ac-
tivemq::transport::failover::FailoverTransport::getInitialReconnectDelay ()
const [inline]`

6.357.2.7 `int activemq::transport::failover::FailoverTransport::getMaxCacheSize ()
const [inline]`

6.357.2.8 `int ac-
tivemq::transport::failover::FailoverTransport::getMaxReconnectAttempts ()
const [inline]`

6.357.2.9 `long long ac-
tivemq::transport::failover::FailoverTransport::getMaxReconnectDelay ()
const [inline]`

6.357.2.10 `long long ac-
tivemq::transport::failover::FailoverTransport::getReconnectDelay ()
const [inline]`

6.357.2.11 `virtual std::string ac-
tivemq::transport::failover::FailoverTransport::getRemoteAddress ()
const [virtual]`

Returns

the remote address for this connection

6.357.2.12 `int activemq::transport::failover::FailoverTransport::getStartupMaxReconnectAttempts () const [inline]`

6.357.2.13 `long long activemq::transport::failover::FailoverTransport::getTimeout () const [inline]`

6.357.2.14 `virtual TransportListener* activemq::transport::failover::FailoverTransport::getTransportListener () const [virtual]`

Gets the observer of asynchronous exceptions from this transport.

Returns

The listener of transport events.

6.357.2.15 `void activemq::transport::failover::FailoverTransport::handleTransportFailure (const decaf::lang::Exception & error) throw (decaf::lang::Exception) [protected]`

Called when this class' **TransportListener** (p. 3643) is notified of a Failure.

Parameters

error - The CMS Exception that was thrown.

Exceptions

Exception if an error occurs.

6.357.2.16 `bool activemq::transport::failover::FailoverTransport::isBackup () const [inline]`

6.357.2.17 `virtual bool activemq::transport::failover::FailoverTransport::isClosed () const [inline, virtual]`

Has the **Transport** (p. 3629) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3629)

6.357.2.18 `virtual bool activemq::transport::failover::FailoverTransport::isConnected () const [inline, virtual]`

Is the **Transport** (p. 3629) Connected to its Broker.

Returns

true if a connection has been made.

6.357.2.19 `virtual bool activemq::transport::failover::FailoverTransport::isFaultTolerant () const [inline, virtual]`

Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3629) is fault tolerant.

6.357.2.20 `bool activemq::transport::failover::FailoverTransport::isInitialized () const [inline]`

Returns true if the **Transport** (p. 3629) has been initialized by a BrokerInfo command.

Returns

true if the **Transport** (p. 3629) has been initialized by a BrokerInfo command.

6.357.2.21 `virtual bool activemq::transport::failover::FailoverTransport::isPending () const [virtual]`

Returns

true if there is a need for the iterate method to be called by this classes task runner.

Implements **activemq::threads::CompositeTask** (p. 1132).

6.357.2.22 `bool activemq::transport::failover::FailoverTransport::isRandomize () const [inline]`

6.357.2.23 `bool activemq::transport::failover::FailoverTransport::isTrackMessages () const [inline]`

6.357.2.24 `bool activemq::transport::failover::FailoverTransport::isTrackTransactionProducers () const [inline]`

6.357.2.25 `bool activemq::transport::failover::FailoverTransport::isUseExponentialBackOff () const [inline]`

6.357.2.26 `virtual bool activemq::transport::failover::FailoverTransport::iterate () [virtual]`

Performs the actual Reconnect operation for the **FailoverTransport** (p. 1752), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.

Returns

false to indicate a connection, true to indicate it needs to try again.

Implements **activemq::threads::Task** (p. 3495).

6.357.2.27 `virtual Transport* activemq::transport::failover::FailoverTransport::narrow (const std::type_info & typeId) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3629) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

typeId - The type_info of the Object we are searching for.

Returns

the requested Object. or NULL if its not in this chain.

References `activemq::transport::Transport::narrow()`.

6.357.2.28 `virtual void activemq::transport::failover::FailoverTransport::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

6.357.2.29 `void activemq::transport::failover::FailoverTransport::reconnect ()`

Indicates that the **Transport** (p. 3629) needs to reconnect to another URI in its list.

6.357.2.30 `virtual void activemq::transport::failover::FailoverTransport::reconnect (const decaf::net::URI & uri) throw (decaf::io::IOException) [virtual]`

reconnect to another location

Parameters

uri

Exceptions

IOException on failure of if not supported

6.357.2.31 `virtual void activemq::transport::failover::FailoverTransport::removeURI (const List< URI > & uris) [virtual]`

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3629) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3629) should result in that **Transport** (p. 3629) being disposed of.

Parameters

uris The new URIs to remove to the set this composite maintains.

6.357.2.32 `virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

command the command to be sent.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

6.357.2.33 `virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

command - The command to be sent.

timeout - The time to wait for this response.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

6.357.2.34 void activemq::transport::failover::FailoverTransport::restoreTransport (const Pointer< Transport > & *transport*) throw (decaf::io::IOException) [protected]

Given a **Transport** (p. 3629) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.

Parameters

transport The new **Transport** (p. 3629) connected to the Broker.

Exceptions

IOException if an errors occurs while restoring the old state.

6.357.2.35 void activemq::transport::failover::FailoverTransport::setBackOffMultiplier (long long *value*) [inline]

6.357.2.36 void activemq::transport::failover::FailoverTransport::setBackup (bool *value*) [inline]

6.357.2.37 void activemq::transport::failover::FailoverTransport::setBackupPoolSize (int *value*) [inline]

6.357.2.38 void activemq::transport::failover::FailoverTransport::setConnectionInterruptProcessingComplete (const Pointer< commands::ConnectionId > & *connectionId*)

6.357.2.39 void activemq::transport::failover::FailoverTransport::setInitialized (bool *value*) [inline]

Sets the initialized state of this **Transport** (p. 3629) to true.

Parameters

value - true if this **Transport** (p. 3629) has been initialized.

- 6.357.2.40** `void activemq::transport::failover::FailoverTransport::setInitialReconnectDelay (long long value) [inline]`
- 6.357.2.41** `void activemq::transport::failover::FailoverTransport::setMaxCacheSize (int value) [inline]`
- 6.357.2.42** `void activemq::transport::failover::FailoverTransport::setMaxReconnectAttempts (int value) [inline]`
- 6.357.2.43** `void activemq::transport::failover::FailoverTransport::setMaxReconnectDelay (long long value) [inline]`
- 6.357.2.44** `void activemq::transport::failover::FailoverTransport::setRandomize (bool value) [inline]`
- 6.357.2.45** `void activemq::transport::failover::FailoverTransport::setReconnectDelay (long long value) [inline]`
- 6.357.2.46** `void activemq::transport::failover::FailoverTransport::setStartupMaxReconnectAttempts (int value) [inline]`
- 6.357.2.47** `void activemq::transport::failover::FailoverTransport::setTimeout (long long value) [inline]`
- 6.357.2.48** `void activemq::transport::failover::FailoverTransport::setTrackMessages (bool value) [inline]`
- 6.357.2.49** `void activemq::transport::failover::FailoverTransport::setTrackTransactionProducers (bool value) [inline]`
- 6.357.2.50** `virtual void activemq::transport::failover::FailoverTransport::setTransportListener (TransportListener * listener) [virtual]`

Sets the observer of asynchronous events from this transport.

Parameters

listener the listener of transport events.

6.357.2.51 void activemq::transport::failover::FailoverTransport::setUseExponentialBackOff (bool *value*) [inline]

6.357.2.52 virtual void activemq::transport::failover::FailoverTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat *AMQCPP_UNUSED*) [inline, virtual]

Sets the WireFormat instance to use.

Parameters

wireFormat The WireFormat the object used to encode / decode commands.

6.357.2.53 virtual void activemq::transport::failover::FailoverTransport::start () throw (decaf::io::IOException) [virtual]

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions

IOException if an error occurs or if this transport has already been closed.

6.357.2.54 virtual void activemq::transport::failover::FailoverTransport::stop () throw (decaf::io::IOException) [virtual]

Stop the Transport (p. 3629).

Exceptions

IOException if an error occurs while stopping the Transport (p. 3629).

6.357.3 Friends And Related Function Documentation

6.357.3.1 friend class FailoverTransportListener [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/FailoverTransport.h

6.358 activemq::transport::failover::FailoverTransportFactory Class Reference

Creates an instance of a FailoverTransport (p. 1752).

```
#include <src/main/activemq/transport/failover/FailoverTransportFactory.h>
```

Inheritance diagram for `activemq::transport::failover::FailoverTransportFactory`:

Public Member Functions

- virtual `~FailoverTransportFactory` ()
- virtual `Pointer< Transport > create` (const `decaf::net::URI` &location) throw (exceptions::ActiveMQException)
*Creates a fully configured **Transport** (p. 3629) instance which could be a chain of filters and transports.*
- virtual `Pointer< Transport > createComposite` (const `decaf::net::URI` &location) throw (exceptions::ActiveMQException)
*Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual `Pointer< Transport > doCreateComposite` (const `decaf::net::URI` &location, const `decaf::util::Properties` &properties) throw (exceptions::ActiveMQException)
*Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.*

6.358.1 Detailed Description

Creates an instance of a `FailoverTransport` (p. 1752).

Since

3.0

6.358.2 Constructor & Destructor Documentation

- 6.358.2.1** virtual `activemq::transport::failover::FailoverTransportFactory::~~FailoverTransportFactory` () [inline, virtual]

6.358.3 Member Function Documentation

- 6.358.3.1** virtual `Pointer<Transport> activemq::transport::failover::FailoverTransportFactory::create` (const `decaf::net::URI` & *location*) throw (exceptions::ActiveMQException) [virtual]

Creates a fully configured `Transport` (p. 3629) instance which could be a chain of filters and transports.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p. 3635).

6.358.3.2 `virtual Pointer<Transport> activemq::transport::failover::FailoverTransportFactory::createComposite (const decaf::net::URI & location) throw (exceptions::ActiveMQException) [virtual]`

Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p. 3635).

6.358.3.3 `virtual Pointer<Transport> activemq::transport::failover::FailoverTransportFactory::doCreateComposite (const decaf::net::URI & location, const decaf::util::Properties & properties) throw (exceptions::ActiveMQException) [protected, virtual]`

Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to.

properties - Properties to apply to the transport.

Returns

Pointer to a new **FailoverTransport** (p. 1752) instance.

Exceptions

ActiveMQException if an error occurs

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/FailoverTransportFactory.h`

6.359 activemq::transport::failover::FailoverTransportListener Class Reference

Utility class used by the **Transport** (p. 3629) to perform the work of responding to events from the active **Transport** (p. 3629).

```
#include <src/main/activemq/transport/failover/FailoverTransportListener.h>
```

Inheritance diagram for `activemq::transport::failover::FailoverTransportListener`:

Public Member Functions

- **FailoverTransportListener** (**FailoverTransport** *parent)
- virtual **~FailoverTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

Event handler for the receipt of a command.

- virtual void **onException** (const **decaf::lang::Exception** &ex)

Event handler for an exception from a command transport.

- virtual void **transportInterrupted** ()

The transport has suffered an interruption from which it hopes to recover.

- virtual void **transportResumed** ()

The transport has resumed after an interruption.

6.359.1 Detailed Description

Utility class used by the **Transport** (p. 3629) to perform the work of responding to events from the active **Transport** (p. 3629).

Since

3.0

6.359.2 Constructor & Destructor Documentation

6.359.2.1 `activemq::transport::failover::FailoverTransportListener::FailoverTransportListener (FailoverTransport * parent)`

6.359.2.2 `virtual
activemq::transport::failover::FailoverTransportListener::~~FailoverTransportListener
() [virtual]`

6.359.3 Member Function Documentation

6.359.3.1 `virtual void ac-
tivemq::transport::failover::FailoverTransportListener::onCommand (
const Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3629) deletes the command upon receipt.

Parameters

command the received command object.

Implements **activemq::transport::TransportListener** (p. 3644).

6.359.3.2 `virtual void ac-
tivemq::transport::failover::FailoverTransportListener::onException (
const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters

ex The exception.

Implements **activemq::transport::TransportListener** (p. 3645).

6.359.3.3 `virtual void ac-
tivemq::transport::failover::FailoverTransportListener::transportInterrupted
() [virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 3645).

6.359.3.4 `virtual void ac-
tivemq::transport::failover::FailoverTransportListener::transportResumed
() [virtual]`

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 3645).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/FailoverTransportListener.h`

6.360 decaf::io::FileDescriptor Class Reference

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

```
#include <src/main/decaf/io/FileDescriptor.h>
```

Inheritance diagram for `decaf::io::FileDescriptor`:

Public Member Functions

- **FileDescriptor** ()
- virtual **~FileDescriptor** ()
- void **sync** ()

*Force any/all buffered data for this **FileDescriptor** (p. 1768) to be flushed to the underlying device.*

- bool **valid** ()

Indicates whether the File Descriptor is valid.

Static Public Attributes

- static **FileDescriptor in**

A handle to the standard input stream.

- static **FileDescriptor out**

A handle to the standard output stream.

- static **FileDescriptor err**

A handle to the standard error stream.

Protected Member Functions

- **FileDescriptor** (long value, bool **readonly**)

Protected Attributes

- long **descriptor**
- bool **readonly**

6.360.1 Detailed Description

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

Since

1.0

6.360.2 Constructor & Destructor Documentation

6.360.2.1 `decaf::io::FileDescriptor::FileDescriptor (long value, bool readonly)`
[protected]

6.360.2.2 `decaf::io::FileDescriptor::FileDescriptor ()`

6.360.2.3 `virtual decaf::io::FileDescriptor::~~FileDescriptor ()` [virtual]

6.360.3 Member Function Documentation

6.360.3.1 `void decaf::io::FileDescriptor::sync ()`

Force any/all buffered data for this **FileDescriptor** (p. 1768) to be flushed to the underlying device.

This method blocks until all data is flushed to the underlying device and is used to place the device into a known state. In the case of data that is buffered at a higher level such as a **BufferedOutputStream** (p. 867) the stream must first be flushed before this method can force the data to be sent to the output device.

6.360.3.2 `bool decaf::io::FileDescriptor::valid ()`

Indicates whether the File Descriptor is valid.

Returns

true for a valid descriptor such as open socket or file, false otherwise.

6.360.4 Field Documentation

6.360.4.1 `long decaf::io::FileDescriptor::descriptor` [protected]

6.360.4.2 `FileDescriptor decaf::io::FileDescriptor::err` [static]

A handle to the standard error stream.

Usually, this file descriptor is not used directly, but rather via the output stream known as `System::err`.

6.360.4.3 `FileDescriptor decaf::io::FileDescriptor::in` [static]

A handle to the standard input stream.

Usually, this file descriptor is not used directly, but rather via the input stream known as `System::in`.

6.360.4.4 `FileDescriptor` `decaf::io::FileDescriptor::out` [static]

A handle to the standard output stream.

Usually, this file descriptor is not used directly, but rather via the output stream known as `System::out`.

6.360.4.5 `bool` `decaf::io::FileDescriptor::readonly` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FileDescriptor.h`

6.361 `decaf::util::logging::Filter` Class Reference

A **Filter** (p. 1770) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

```
#include <src/main/decaf/util/logging/Filter.h>
```

Public Member Functions

- virtual `~Filter` ()
- virtual `bool` `isLoggable` (const **LogRecord** &record) const =0

Check if a given log record should be published.

6.361.1 Detailed Description

A **Filter** (p. 1770) can be used to provide fine grain control over what is logged, beyond the control provided by log levels. Each **Logger** (p. 2237) and each **Handler** (p. 1852) can have a filter associated with it. The **Logger** (p. 2237) or **Handler** (p. 1852) will call the `isLoggable` method to check if a given **LogRecord** (p. 2261) should be published. If `isLoggable` returns false, the **LogRecord** (p. 2261) will be discarded.

6.361.2 Constructor & Destructor Documentation

6.361.2.1 virtual `decaf::util::logging::Filter::~Filter` () [inline, virtual]

6.361.3 Member Function Documentation

6.361.3.1 virtual `bool` `decaf::util::logging::Filter::isLoggable` (const **LogRecord** & *record*) const [pure virtual]

Check if a given log record should be published.

Parameters

record the LogRecord (p. 2261) to check.

Returns

true if the record is loggable.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Filter.h**

6.362 decaf::io::FilterInputStream Class Reference

A **FilterInputStream** (p. 1771) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

```
#include <src/main/decaf/io/FilterInputStream.h>
```

Inheritance diagram for decaf::io::FilterInputStream:

Public Member Functions

- **FilterInputStream** (**InputStream** **inputStream*, bool *own*=false)

*Constructor to create a wrapped **InputStream** (p. 1909).*

- virtual ~**FilterInputStream** ()
- virtual int **available** () const throw (decaf::io::IOException)

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

***IOException** (p. 2003) if an I/O error occurs.*

- virtual void **close** () throw (decaf::io::IOException)

*Closes the **InputStream** (p. 1909) freeing any resources that might have been acquired during the lifetime of this stream.*

The default implementation of this method does nothing.

- virtual long long **skip** (long long *num*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

*Skips over and discards *n* bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.*

The `skip` method of ***InputStream*** (p. 1909) creates a byte array and then repeatedly reads into it until `num` bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

`num` The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

- virtual void **`mark`** (int `readLimit`)

Marks the current position in the stream. A subsequent call to the `reset` method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the `reset` method is called so long the `readLimit` is not reached.

Calling `mark` on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

`readLimit` The max bytes read before marked position is invalid.

- virtual void **`reset`** () throw (`decaf::io::IOException`)

Repositions this stream to the position at the time the `mark` method was last called on this input stream.

If the method `markSupported` returns `true`, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since `mark` was last called is larger than the argument to `mark` at that last call, then an ***IOException*** (p. 2003) might be thrown. * If such an ***IOException*** (p. 2003) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to `mark` (or since the start of the file, if `mark` has not been called) will be resupplied to subsequent callers of the `read` method, followed by any bytes that otherwise would have been the next input data as of the time of the call to `reset`.

If the method `markSupported` returns `false`, then: * The call to `reset` may throw an ***IOException*** (p. 2003). * If an ***IOException*** (p. 2003) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the `read` method depend on the particular type of the input stream.

The default implementation of this method throws an ***IOException*** (p. 2003).

Exceptions

IOException (p. 2003) if an I/O error occurs.

- virtual bool **`markSupported`** () const

Determines if this input stream supports the `mark` and `reset` methods.

Whether or not `mark` and `reset` are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns `false`.

Returns

`true` if this stream instance supports marks

Protected Member Functions

- virtual int **`doReadByte`** () throw (`decaf::io::IOException`)

- virtual int **doReadArray** (unsigned char *buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)
- virtual bool **isClosed** () const

Protected Attributes

- **InputStream * inputStream**
- bool **own**
- volatile bool **closed**

6.362.1 Detailed Description

A **FilterInputStream** (p.1771) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality. The class **FilterInputStream** (p.1771) itself simply overrides all methods of **InputStream** (p.1909) with versions that pass all requests to the contained input stream. Subclasses of **FilterInputStream** (p.1771) may further override some of these methods and may also provide additional methods and fields.

6.362.2 Constructor & Destructor Documentation

6.362.2.1 decaf::io::FilterInputStream::FilterInputStream (**InputStream** * *inputStream*, bool *own* = *false*)

Constructor to create a wrapped **InputStream** (p.1909).

Parameters

inputStream The stream to wrap and filter.

own Indicates if we own the stream object, defaults to false.

6.362.2.2 virtual decaf::io::FilterInputStream::~FilterInputStream () [virtual]

6.362.3 Member Function Documentation

6.362.3.1 virtual int decaf::io::FilterInputStream::available () const throw (decaf::io::IOException) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented from `decaf::io::InputStream` (p. 1911).

Reimplemented in `decaf::io::BufferedInputStream` (p. 864), `decaf::io::PushbackInputStream` (p. 2943), and `decaf::util::zip::InflaterInputStream` (p. 1906).

6.362.3.2 `virtual void decaf::io::FilterInputStream::close () throw (decaf::io::IOException)` [virtual]

Closes the `InputStream` (p. 1909) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from `decaf::io::InputStream` (p. 1912).

Reimplemented in `decaf::io::BufferedInputStream` (p. 864), and `decaf::util::zip::InflaterInputStream` (p. 1906).

6.362.3.3 `virtual int decaf::io::FilterInputStream::doReadArray (unsigned char * buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)` [protected, virtual]

Reimplemented from `decaf::io::InputStream` (p. 1912).

6.362.3.4 `virtual int decaf::io::FilterInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)` [protected, virtual]

Reimplemented from `decaf::io::InputStream` (p. 1912).

Reimplemented in `activemq::io::LoggingInputStream` (p. 2250), `decaf::io::BufferedInputStream` (p. 865), `decaf::io::PushbackInputStream` (p. 2943), `decaf::util::zip::CheckedInputStream` (p. 1057), and `decaf::util::zip::InflaterInputStream` (p. 1906).

6.362.3.5 `virtual int decaf::io::FilterInputStream::doReadByte () throw (decaf::io::IOException)` [protected, virtual]

Implements `decaf::io::InputStream` (p. 1912).

Reimplemented in `activemq::io::LoggingInputStream` (p. 2251), `decaf::io::BufferedInputStream` (p. 865), `decaf::io::PushbackInputStream` (p. 2943), `decaf::util::zip::CheckedInputStream` (p. 1057), and `decaf::util::zip::InflaterInputStream` (p. 1906).

6.362.3.6 `virtual bool decaf::io::FilterInputStream::isClosed () const`
[protected, virtual]

Returns

true if this stream has been closed.

6.362.3.7 `virtual void decaf::io::FilterInputStream::mark (int readLimit)`
[virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit The max bytes read before marked position is invalid.

Reimplemented from `decaf::io::InputStream` (p. 1913).

Reimplemented in `decaf::io::BufferedInputStream` (p. 865), `decaf::io::PushbackInputStream` (p. 2944), and `decaf::util::zip::InflaterInputStream` (p. 1907).

6.362.3.8 `virtual bool decaf::io::FilterInputStream::markSupported () const`
[virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented from `decaf::io::InputStream` (p. 1913).

Reimplemented in `decaf::io::BufferedInputStream` (p. 865), `decaf::io::PushbackInputStream` (p. 2944), and `decaf::util::zip::InflaterInputStream` (p. 1907).

6.362.3.9 `virtual void decaf::io::FilterInputStream::reset () throw (decaf::io::IOException)` [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method `markSupported` returns true, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since `mark` was last called is larger than the argument to `mark` at that last call, then an **IOException** (p. 2003) might be thrown. * If such an **IOException** (p. 2003) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to `mark` (or since the start of the file, if `mark` has not been called) will be resupplied to subsequent callers of the `read` method, followed by any bytes that otherwise would have been the next input data as of the time of the call to `reset`.

If the method `markSupported` returns false, then: * The call to `reset` may throw an **IOException** (p. 2003). * If an **IOException** (p. 2003) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the `read` method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2003).

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1916).

Reimplemented in **decaf::io::BufferedInputStream** (p. 865), **decaf::io::PushbackInputStream** (p. 2944), and **decaf::util::zip::InflaterInputStream** (p. 1907).

6.362.3.10 `virtual long long decaf::io::FilterInputStream::skip
(long long num) throw (decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException)` [virtual]

Skips over and discards `n` bytes of data from this input stream.

The `skip` method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before `n` bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The `skip` method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until `num` bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1917).

Reimplemented in **decaf::io::BufferedInputStream** (p. 866), **decaf::io::PushbackInputStream** (p. 2945), **decaf::util::zip::CheckedInputStream** (p. 1057), and **decaf::util::zip::InflaterInputStream** (p. 1908).

6.362.4 Field Documentation

6.362.4.1 `volatile bool decaf::io::FilterInputStream::closed` [protected]

6.362.4.2 `InputStream* decaf::io::FilterInputStream::inputStream` [protected]

6.362.4.3 `bool decaf::io::FilterInputStream::own` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FilterInputStream.h`

6.363 decaf::io::FilterOutputStream Class Reference

This class is the superclass of all classes that filter output streams.

```
#include <src/main/decaf/io/FilterOutputStream.h>
```

Inheritance diagram for decaf::io::FilterOutputStream:

Public Member Functions

- **FilterOutputStream** (**OutputStream** *outputStream, bool own=false)

Constructor, creates a wrapped output stream.

- virtual **~FilterOutputStream** ()
- virtual void **flush** () throw (decaf::io::IOException)

The default implementation of this method does nothing.

- virtual void **close** () throw (decaf::io::IOException)

The default implementation of this method does nothing.

- virtual std::string **toString** () const

Output a String representation of this object.

The default version of this method just prints the Class Name.

Returns

a string representation of the object.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw (decaf::io::IOException)
- virtual void **doWriteArray** (const unsigned char *buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual bool **isClosed** () const

Protected Attributes

- **OutputStream** * **outputStream**
- bool **own**
- volatile bool **closed**

6.363.1 Detailed Description

This class is the superclass of all classes that filter output streams. These streams sit on top of an already existing output stream (the underlying output stream) which it uses as its basic sink of data, but possibly transforming the data along the way or providing additional functionality.

The class **FilterOutputStream** (p. 1777) itself simply overrides all methods of **OutputStream** (p. 2718) with versions that pass all requests to the underlying output stream. Subclasses of **FilterOutputStream** (p. 1777) may further override some of these methods as well as provide additional methods and fields.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 1909) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

```
DataOutputStream (p. 1473) os = new DataOutputStream (p. 1473)( new Output-Stream() (p. 2720), true )
```

6.363.2 Constructor & Destructor Documentation

6.363.2.1 `decaf::io::FilterOutputStream::FilterOutputStream (OutputStream * outputStream, bool own = false)`

Constructor, creates a wrapped output stream.

Parameters

outputStream the **OutputStream** (p. 2718) to wrap

own If true, this object will control the lifetime of the output stream that it encapsulates.

6.363.2.2 `virtual decaf::io::FilterOutputStream::~~FilterOutputStream ()` [virtual]

6.363.3 Member Function Documentation

6.363.3.1 `virtual void decaf::io::FilterOutputStream::close () throw (decaf::io::IOException)` [virtual]

The default implementation of this method does nothing.

The close method of **FilterOutputStream** (p. 1777) calls its flush method, and then calls the close method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2720).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 1607).

6.363.3.2 `virtual void decaf::io::FilterOutputStream::doWriteArray (const unsigned char * buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)` [protected, virtual]

Reimplemented from `decaf::io::OutputStream` (p. 2721).

Reimplemented in `decaf::io::BufferedOutputStream` (p. 868).

6.363.3.3 `virtual void decaf::io::FilterOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)` [protected, virtual]

Reimplemented from `decaf::io::OutputStream` (p. 2721).

Reimplemented in `activemq::io::LoggingOutputStream` (p. 2252), `decaf::io::BufferedOutputStream` (p. 868), `decaf::io::DataOutputStream` (p. 1475), `decaf::util::zip::CheckedOutputStream` (p. 1059), and `decaf::util::zip::DeflaterOutputStream` (p. 1607).

6.363.3.4 `virtual void decaf::io::FilterOutputStream::doWriteByte (unsigned char value) throw (decaf::io::IOException)` [protected, virtual]

Implements `decaf::io::OutputStream` (p. 2721).

Reimplemented in `activemq::io::LoggingOutputStream` (p. 2252), `decaf::io::BufferedOutputStream` (p. 868), `decaf::io::DataOutputStream` (p. 1475), `decaf::util::zip::CheckedOutputStream` (p. 1059), and `decaf::util::zip::DeflaterOutputStream` (p. 1607).

6.363.3.5 `virtual void decaf::io::FilterOutputStream::flush () throw (decaf::io::IOException)` [virtual]

The default implementation of this method does nothing.

The flush method of `FilterOutputStream` (p. 1777) calls the flush method of its underlying output stream.

Reimplemented from `decaf::io::OutputStream` (p. 2721).

Reimplemented in `decaf::io::BufferedOutputStream` (p. 868).

6.363.3.6 `virtual bool decaf::io::FilterOutputStream::isClosed () const` [protected, virtual]

Returns

true if this stream has been closed.

6.363.3.7 virtual std::string decaf::io::FilterOutputStream::toString () const [virtual]

Output a String representation of this object.

The default version of this method just prints the Class Name.

Returns

a string representation of the object.

The toString method of **FilterOutputStream** (p. 1777) calls the toString method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2722).

6.363.4 Field Documentation

6.363.4.1 volatile bool decaf::io::FilterOutputStream::closed [protected]

6.363.4.2 OutputStream* decaf::io::FilterOutputStream::outputStream [protected]

6.363.4.3 bool decaf::io::FilterOutputStream::own [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**FilterOutputStream.h**

6.364 decaf::lang::Float Class Reference

```
#include <src/main/decaf/lang/Float.h>
```

Inheritance diagram for decaf::lang::Float:

Public Member Functions

- **Float** (float value)
- **Float** (double value)
- **Float** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Float** ()
- virtual int **compareTo** (const **Float** &f) const
*Compares this **Float** (p. 1780) instance with another.*
- bool **equals** (const **Float** &f) const
- virtual bool **operator==** (const **Float** &f) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Float** &f) const

Compares this object to another and returns true if this object is considered to be less than the one passed.

- virtual int **compareTo** (const float &f) const

*Compares this **Float** (p. 1780) instance with another.*

- bool **equals** (const float &f) const
- virtual bool **operator==** (const float &f) const

Compares equality between this object and the one passed.

- virtual bool **operator<** (const float &f) const

Compares this object to another and returns true if this object is considered to be less than the one passed.

- std::string **toString** () const
- virtual double **doubleValue** () const

Answers the double value which the receiver represents.

- virtual float **floatValue** () const

Answers the float value which the receiver represents.

- virtual unsigned char **byteValue** () const

Answers the byte value which the receiver represents.

- virtual short **shortValue** () const

Answers the short value which the receiver represents.

- virtual int **intValue** () const

Answers the int value which the receiver represents.

- virtual long long **longValue** () const

Answers the long value which the receiver represents.

- bool **isInfinite** () const
- bool **isNaN** () const

Static Public Member Functions

- static int **compare** (float f1, float f2)

Compares the two specified double values.

- static int **floatToIntBits** (float value)

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.

- static int **floatToRawIntBits** (float value)

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.

- static float **intBitsToFloat** (int bits)

Returns the float value corresponding to a given bit representation.

- static bool **isInfinite** (float value)
- static bool **isNaN** (float value)
- static float **parseFloat** (const std::string &value) throw (exceptions::NumberFormatException)

*Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1780).*

- static std::string **toHexString** (float value)

Returns a hexadecimal string representation of the float argument.

- static std::string **toString** (float value)

Returns a string representation of the float argument.

- static **Float valueOf** (float value)

*Returns a **Float** (p. 1780) instance representing the specified float value.*

- static **Float valueOf** (const std::string &value) throw (exceptions::NumberFormatException)

*Returns a **Float** (p. 1780) instance that wraps a primitive float which is parsed from the string value passed.*

Static Public Attributes

- static const int **SIZE** = 32

The size in bits of the primitive int type.

- static const float **MAX_VALUE**

The maximum value that the primitive type can hold.

- static const float **MIN_VALUE**

The minimum value that the primitive type can hold.

- static const float **NaN**

*Constant for the Not a **Number** (p. 2653) Value.*

- static const float **POSITIVE_INFINITY**

Constant for Positive Infinity.

- static const float **NEGATIVE_INFINITY**

Constant for Negative Infinity.

6.364.1 Constructor & Destructor Documentation

6.364.1.1 decaf::lang::Float::Float (float value)

Parameters

value - the primitive type to wrap

6.364.1.2 decaf::lang::Float::Float (double *value*)

Parameters

value - the primitive type to wrap

6.364.1.3 decaf::lang::Float::Float (const std::string & *value*) throw (exceptions::NumberFormatException)

Parameters

value - the string to convert to a primitive type to wrap

6.364.1.4 virtual decaf::lang::Float::~~Float () [inline, virtual]

6.364.2 Member Function Documentation

6.364.2.1 virtual unsigned char decaf::lang::Float::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

6.364.2.2 static int decaf::lang::Float::compare (float *f1*, float *f2*) [static]

Compares the two specified double values.

The sign of the integer value returned is the same as that of the integer that would be returned by the call: `Float(f1).compareTo(Float(f2))`

Parameters

f1 - the first double to compare

f2 - the second double to compare

Returns

the value 0 if *d1* is numerically equal to *f2*; a value less than 0 if *f1* is numerically less than *f2*; and a value greater than 0 if *f1* is numerically greater than *f2*.

6.364.2.3 virtual int decaf::lang::Float::compareTo (const float & *f*) const [virtual]

Compares this **Float** (p. 1780) instance with another.

Parameters

f - the **Float** (p. 1780) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **float** > (p. 1126).

6.364.2.4 `virtual int decaf::lang::Float::compareTo (const Float & f) const`
[virtual]

Compares this **Float** (p. 1780) instance with another.

Parameters

f - the **Float** (p. 1780) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.364.2.5 `virtual double decaf::lang::Float::doubleValue () const` [inline,
virtual]

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.364.2.6 `bool decaf::lang::Float::equals (const Float & f) const` [inline]

Parameters

f - the **Float** (p. 1780) object to compare against.

Returns

true if the two **Float** (p. 1780) Objects have the same value.

6.364.2.7 `bool decaf::lang::Float::equals (const float & f) const` [inline,
virtual]

Parameters

f - the **Float** (p. 1780) object to compare against.

Returns

true if the two **Float** (p. 1780) Objects have the same value.

Implements **decaf::lang::Comparable**< **float** > (p. 1126).

6.364.2.8 static int decaf::lang::Float::floatToIntBits (float *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.

Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is 0x7c000000.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1786) method, will produce a floating-point value the same as the argument to floatToIntBits (except all NaN values are collapsed to a single "canonical" NaN value).

Parameters

value - the float to convert to int bits

Returns

the int that holds the float's value

6.364.2.9 static int decaf::lang::Float::floatToRawIntBits (float *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.

Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is the integer representing the actual NaN value. Unlike the floatToIntBits method, intToRawIntBits does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1786) method, will produce a floating-point value the same as the argument to floatToRawIntBits.

Parameters

value The float to convert to a raw int.

Returns

the raw int value of the float

6.364.2.10 virtual float decaf::lang::Float::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.364.2.11 static float decaf::lang::Float::intBitsToFloat (int *bits*) [static]

Returns the float value corresponding to a given bit representation.

The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "single format" bit layout.

If the argument is 0x7f800000, the result is positive infinity. If the argument is 0xff800000, the result is negative infinity. If the argument is any value in the range 0x7f800001 through 0x7fffffff or in the range 0xff800001 through 0xffffffff, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Float::floatToRawIntBits** (p. 1785) method.

Parameters

bits - the bits of the float encoded as a float

Returns

a new float created from the int bits.

6.364.2.12 virtual int decaf::lang::Float::intValue () const [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.364.2.13 bool decaf::lang::Float::isInfinite () const**Returns**

true if the float is equal to positive infinity.

6.364.2.14 static bool decaf::lang::Float::isInfinite (float *value*) [static]**Parameters**

value - The float to check.

Returns

true if the float is equal to infinity.

6.364.2.15 bool decaf::lang::Float::isNaN () const**Returns**

true if the float is equal to NaN.

6.364.2.16 `static bool decaf::lang::Float::isNaN (float value) [static]`

Parameters

value - The float to check.

Returns

true if the float is equal to NaN.

6.364.2.17 `virtual long long decaf::lang::Float::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.364.2.18 `virtual bool decaf::lang::Float::operator< (const float & f) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

f - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< float >` (p. 1127).

6.364.2.19 `virtual bool decaf::lang::Float::operator< (const Float & f) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

f - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.364.2.20 `virtual bool decaf::lang::Float::operator==(const Float & f) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

f - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.364.2.21 `virtual bool decaf::lang::Float::operator==(const float & f) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

f - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< float >` (p. 1127).

6.364.2.22 `static float decaf::lang::Float::parseFloat (const std::string & value)`
`throw (exceptions::NumberFormatException)` [static]

Returns a new float initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Float** (p. 1780).

Parameters

value - the string to parse

Returns

a float parsed from the string

Exceptions

NumberFormatException

6.364.2.23 `virtual short decaf::lang::Float::shortValue () const` [inline,
virtual]

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

6.364.2.24 `static std::string decaf::lang::Float::toHexString (float value)`
[static]

Returns a hexadecimal string representation of the float argument.

All characters mentioned below are ASCII characters.

* If the argument is NaN, the result is the string "NaN". * Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a float value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p.1954) on the exponent value. o If m is a float value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

Parameters

value - The float to convert to a string

Returns

the Hex formatted float string.

6.364.2.25 `std::string decaf::lang::Float::toString () const`**Returns**

this **Float** (p.1780) Object as a **String** (p.3427) Representation

6.364.2.26 `static std::string decaf::lang::Float::toString (float value)` [static]

Returns a string representation of the float argument.

All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0". o If m is greater than or equal to 10⁻³ but less than 10⁷, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of m. o If m is less than 10⁻³ or greater than or equal to 10⁷, then it is represented in so-called "computerized

scientific notation." Let n be the unique integer such that $10^n \leq m < 10^{n+1}$; then let a be the mathematically exact quotient of m and 10^n so that $1 \leq a < 10$. The magnitude is then represented as the integer part of a , as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of a , followed by the letter 'E', followed by a representation of n as a decimal integer, as produced by the method **Integer.toString(int)** (p. 1954).

Parameters

value - The float to convert to a string

Returns

the formatted float string.

6.364.2.27 `static Float decaf::lang::Float::valueOf (const std::string & value)
throw (exceptions::NumberFormatException) [static]`

Returns a **Float** (p. 1780) instance that wraps a primitive float which is parsed from the string value passed.

Parameters

value - the string to parse

Returns

a new **Float** (p. 1780) instance wrapping the float parsed from value

Exceptions

NumberFormatException on error.

6.364.2.28 `static Float decaf::lang::Float::valueOf (float value) [static]`

Returns a **Float** (p. 1780) instance representing the specified float value.

Parameters

value - float to wrap

Returns

new **Float** (p. 1780) instance wrapping the primitive value

6.364.3 Field Documentation

6.364.3.1 `const float decaf::lang::Float::MAX_VALUE [static]`

The maximum value that the primitive type can hold.

6.364.3.2 `const float decaf::lang::Float::MIN_VALUE [static]`

The minimum value that the primitive type can hold.

6.364.3.3 `const float decaf::lang::Float::NaN` [static]

Constant for the Not a **N**umber (p. 2653) Value.

6.364.3.4 `const float decaf::lang::Float::NEGATIVE_INFINITY` [static]

Constant for Negative Infinity.

6.364.3.5 `const float decaf::lang::Float::POSITIVE_INFINITY` [static]

Constant for Positive Infinity.

6.364.3.6 `const int decaf::lang::Float::SIZE = 32` [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Float.h`

6.365 `decaf::internal::nio::FloatArrayBuffer` Class Reference

```
#include <src/main/decaf/internal/nio/FloatArrayBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::FloatArrayBuffer`:

Public Member Functions

- **FloatArrayBuffer** (int size, bool readOnly=false) throw (decaf::lang::exceptions::IllegalArgumentException)

*Creates a **FloatArrayBuffer** (p. 1791) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **FloatArrayBuffer** (float *array, int size, int offset, int length, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

*Creates a **FloatArrayBuffer** (p. 1791) object that wraps the given array.*

- **FloatArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

*Creates a byte buffer that wraps the passed **ByteArrayAdapter** and start at the given offset.*

- **FloatArrayBuffer** (const **FloatArrayBuffer** &other)

*Create a **FloatArrayBuffer** (p. 1791) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayAdapter** and when changes are made to that data it is reflected in both.*

- virtual `~FloatArrayBuffer ()`
- virtual `float * array ()` throw (`decaf::lang::exceptions::UnsupportedOperationException`, `decaf::nio::ReadOnlyBufferException`)

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

*the array that backs this **Buffer** (p. 855).*

Exceptions

***ReadOnlyBufferException** (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.*

- virtual `int arrayOffset ()` throw (`decaf::lang::exceptions::UnsupportedOperationException`, `decaf::nio::ReadOnlyBufferException`)

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

***ReadOnlyBufferException** (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.*

- virtual `FloatBuffer * asReadOnlyBuffer ()` const

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only float buffer which the caller then owns.

- virtual `FloatBuffer & compact ()` throw (`decaf::nio::ReadOnlyBufferException`)

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 860) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 859) - 1 is copied to index `n = limit()` (p. 859) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

*a reference to this **FloatBuffer** (p. 1800).*

Exceptions

***ReadOnlyBufferException** (p. 2966) if this buffer is read-only*

- virtual FloatBuffer * **duplicate** ()

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*a new float **Buffer** (p. 855) which the caller owns.*

- virtual float **get** () throw (decaf::nio::BufferUnderflowException)

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the float at the current position.

Exceptions

***BufferUnderflowException** (p. 882) if there no more data to return.*

- virtual float **get** (int index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

Reads the value at the given index.

Parameters

***index** The index in the **Buffer** (p. 855) where the float is to be read*

Returns

the float that is located at the given index

Exceptions

***IndexOutOfBoundsException** if index is not smaller than the buffer's limit*

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

- virtual bool **isReadOnly** () const

- virtual FloatBuffer & **put** (float value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes the given floats into this buffer at the current position, and then increments the position.

Parameters

***value** The floats value to be written.*

Returns

a reference to this buffer.

Exceptions

***BufferOverflowException** (p. 880) if this buffer's current position is not smaller than its limit*

ReadOnlyBufferException (p. 2966) if this buffer is read-only

- virtual FloatBuffer & put (int index, float value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes the given floats into this buffer at the given index.

Parameters

index *The position in the Buffer (p. 855) to write the data.*
value *The floats to write.*

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual FloatBuffer * slice () const

Creates a new FloatBuffer (p. 1800) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create FloatBuffer (p. 1800) which the caller owns.

Protected Member Functions

- virtual void setReadOnly (bool value)

Sets this FloatArrayBuffer (p. 1791) as Read-Only.

6.365.1 Constructor & Destructor Documentation

6.365.1.1 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (int size, bool readOnly = false) throw (decaf::lang::exceptions::IllegalArgumentException)

Creates a **FloatArrayBuffer** (p. 1791) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

size *The size of the array, this is the limit we read and write to.*

readOnly *Boolean indicating if this buffer should be read-only, default as false.*

Exceptions

IllegalArgumentException if the capacity value is negative.

6.365.1.2 `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (float * array, int size, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a **FloatArrayBuffer** (p.1791) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The actual array to wrap.

size The size of the given array.

offset The position that is this buffers start position.

length The limit of how many bytes into the array this Buffer can write.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException if buffer is NULL

IndexOutOfBoundsException if offset is greater than array capacity.

6.365.1.3 `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **FloatArrayBuffer** (p.1791) will be that of the remaining capacity of the passed buffer.

Parameters

array The ByteArrayAdapter to wrap.

offset The position that is this buffers start position.

length The limit of how many bytes into the array this Buffer can write.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException if array is NULL

IndexOutOfBoundsException if offset + length is greater than array size.

6.365.1.4 `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (const FloatArrayBuffer & other)`

Create a **FloatArrayBuffer** (p.1791) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

Parameters

other The **FloatArrayBuffer** (p.1791) this one is to mirror.

6.365.1.5 `virtual decaf::internal::nio::FloatArrayBuffer::~~FloatArrayBuffer ()`
[virtual]

6.365.2 Member Function Documentation

6.365.2.1 `virtual float* decaf::internal::nio::FloatArrayBuffer::array ()`
`throw (decaf::lang::exceptions::UnsupportedOperationException,`
`decaf::nio::ReadOnlyBufferException)` [virtual]

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 855).

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implements `decaf::nio::FloatBuffer` (p. 1803).

6.365.2.2 `virtual int decaf::internal::nio::FloatArrayBuffer::arrayOffset ()`
`throw (decaf::lang::exceptions::UnsupportedOperationException,`
`decaf::nio::ReadOnlyBufferException)` [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implements `decaf::nio::FloatBuffer` (p. 1804).

6.365.2.3 `virtual FloatBuffer* de-`
`caf::internal::nio::FloatArrayBuffer::asReadOnlyBuffer (`
`) const` [virtual]

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only float buffer which the caller then owns.

Implements **decaf::nio::FloatBuffer** (p.1804).

6.365.2.4 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p.860) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p.859) - 1 is copied to index `n = limit()` (p.859) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **FloatBuffer** (p.1800).

Exceptions

ReadOnlyBufferException (p.2966) if this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p.1804).

6.365.2.5 virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::duplicate () [virtual]

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new float **Buffer** (p.855) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p.1805).

6.365.2.6 `virtual float decaf::internal::nio::FloatArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the float at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return.

Implements `decaf::nio::FloatBuffer` (p.1807).

6.365.2.7 `virtual float decaf::internal::nio::FloatArrayBuffer::get (int index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Absolute get method.

Reads the value at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the float is to be read

Returns

the float that is located at the given index

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit

Implements `decaf::nio::FloatBuffer` (p.1806).

6.365.2.8 `virtual bool decaf::internal::nio::FloatArrayBuffer::hasArray () const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements `decaf::nio::FloatBuffer` (p.1807).

6.365.2.9 virtual bool decaf::internal::nio::FloatArrayBuffer::isReadOnly ()
const [inline, virtual]

6.365.2.10 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put
(int *index*, float *value*) throw (lang::exceptions::IndexOutOfBoundsException,
decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given floats into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The floats to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::FloatBuffer** (p. 1808).

6.365.2.11 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put
(float *value*) throw (decaf::nio::BufferOverflowException,
decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given floats into this buffer at the current position, and then increments the position.

Parameters

value The floats value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1807).

6.365.2.12 virtual void decaf::internal::nio::FloatArrayBuffer::setReadOnly (bool
value) [inline, protected, virtual]

Sets this **FloatArrayBuffer** (p. 1791) as Read-Only.

Parameters

value Boolean value, true if this buffer is to be read-only, false otherwise.

6.365.2.13 virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::slice () const [virtual]

Creates a new **FloatBuffer** (p. 1800) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **FloatBuffer** (p. 1800) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1809).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**FloatArrayBuffer.h**

6.366 decaf::nio::FloatBuffer Class Reference

This class defines four categories of operations upon float buffers:

```
#include <src/main/decaf/nio/FloatBuffer.h>
```

Inheritance diagram for decaf::nio::FloatBuffer:

Public Member Functions

- virtual **~FloatBuffer** ()
- virtual std::string **toString** () const
- virtual float * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the float array that backs this buffer (optional operation).
- virtual int **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **FloatBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only float buffer that shares this buffer's content.
- virtual **FloatBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **FloatBuffer** * **duplicate** ()=0

Creates a new float buffer that shares this buffer's content.

- virtual float **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual float **get** (int index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **FloatBuffer & get** (std::vector< float > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **FloatBuffer & get** (float *buffer, int size, int offset, int length) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible float array.
- **FloatBuffer & put** (**FloatBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException)
This method transfers the floats remaining in the given source buffer into this buffer.
- **FloatBuffer & put** (const float *buffer, int size, int offset, int length) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)
This method transfers floats into this buffer from the given source array.
- **FloatBuffer & put** (std::vector< float > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source floats array into this buffer.
- virtual **FloatBuffer & put** (float value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given floats into this buffer at the current position, and then increments the position.
- virtual **FloatBuffer & put** (int index, float value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given floats into this buffer at the given index.
- virtual **FloatBuffer * slice** () const =0
*Creates a new **FloatBuffer** (p.1800) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **FloatBuffer** &value) const
- virtual bool **equals** (const **FloatBuffer** &value) const

- virtual bool **operator==** (const **FloatBuffer** &value) const
- virtual bool **operator<** (const **FloatBuffer** &value) const

Static Public Member Functions

- static **FloatBuffer** * **allocate** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)
Allocates a new Double buffer.
- static **FloatBuffer** * **wrap** (float *array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
*Wraps the passed buffer with a new **FloatBuffer** (p. 1800).*
- static **FloatBuffer** * **wrap** (std::vector< float > &buffer)
*Wraps the passed STL float Vector in a **FloatBuffer** (p. 1800).*

Protected Member Functions

- **FloatBuffer** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)
*Creates a **FloatBuffer** (p. 1800) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.366.1 Detailed Description

This class defines four categories of operations upon float buffers: o Absolute and relative get and put methods that read and write single floats; o Relative bulk get methods that transfer contiguous sequences of floats from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of floats from a float array or some other float buffer into this buffer o Methods for compacting, duplicating, and slicing a float buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing float array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.366.2 Constructor & Destructor Documentation

- #### 6.366.2.1 decaf::nio::FloatBuffer::FloatBuffer (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException) [protected]

Creates a **FloatBuffer** (p. 1800) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity The size and limit of the **Buffer** (p. 855) in floats.

Exceptions

IllegalArgumentException if capacity is negative.

6.366.2.2 virtual decaf::nio::FloatBuffer::~~FloatBuffer () [inline, virtual]

6.366.3 Member Function Documentation

6.366.3.1 static FloatBuffer* decaf::nio::FloatBuffer::allocate (int *capacity*)
throw (decaf::lang::exceptions::IllegalArgumentException) [static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity The size of the Double buffer in floats.

Returns

the **FloatBuffer** (p. 1800) that was allocated, caller owns.

6.366.3.2 virtual float* decaf::nio::FloatBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 855).

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in decaf::internal::nio::FloatArrayBuffer (p. 1796).

6.366.3.3 `virtual int decaf::nio::FloatBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1796).

6.366.3.4 `virtual FloatBuffer* decaf::nio::FloatBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only float buffer which the caller then owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1796).

6.366.3.5 `virtual FloatBuffer& decaf::nio::FloatBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 860) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 859) - 1 is copied to index `n = limit()` (p. 859) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **FloatBuffer** (p. 1800).

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1797).

6.366.3.6 `virtual int decaf::nio::FloatBuffer::compareTo (const FloatBuffer & value) const` [virtual]

6.366.3.7 `virtual FloatBuffer* decaf::nio::FloatBuffer::duplicate ()` [pure virtual]

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new float **Buffer** (p. 855) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1797).

6.366.3.8 `virtual bool decaf::nio::FloatBuffer::equals (const FloatBuffer & value) const` [virtual]

6.366.3.9 `FloatBuffer& decaf::nio::FloatBuffer::get (std::vector< float > buffer) throw (BufferUnderflowException)`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than length floats remaining in this buffer

6.366.3.10 `virtual float decaf::nio::FloatBuffer::get (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the float is to be read

Returns

the float that is located at the given index

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1798).

6.366.3.11 `FloatBuffer& decaf::nio::FloatBuffer::get (float * buffer, int size, int offset, int length) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)`

Relative bulk get method.

This method transfers floats from this buffer into the given destination array. If there are fewer floats remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 860), then no bytes are transferred and a **BufferUnderflowException** (p. 882) is thrown.

Otherwise, this method copies `length` floats from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters

buffer The pointer to an allocated buffer to fill.

size The size of the passed in buffer.

offset The position in the buffer to start filling.

length The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than `length` floats remaining in this buffer

NullPointerException if the passed buffer is null.

IndexOutOfBoundsException if the preconditions of `size`, `offset`, or `length` are not met.

6.366.3.12 `virtual float decaf::nio::FloatBuffer::get () throw (BufferUnderflowException)` [pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the float at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return.

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1798).

6.366.3.13 `virtual bool decaf::nio::FloatBuffer::hasArray () const` [pure virtual]

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1798).

6.366.3.14 `virtual bool decaf::nio::FloatBuffer::operator< (const FloatBuffer & value) const` [virtual]

6.366.3.15 `virtual bool decaf::nio::FloatBuffer::operator== (const FloatBuffer & value) const` [virtual]

6.366.3.16 `virtual FloatBuffer& decaf::nio::FloatBuffer::put (float value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given floats into this buffer at the current position, and then increments the position.

Parameters

value The floats value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1799).

6.366.3.17 `virtual FloatBuffer& decaf::nio::FloatBuffer::put (int index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]`

Writes the given floats into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.
value The floats to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1799).

6.366.3.18 `FloatBuffer& decaf::nio::FloatBuffer::put (const float * buffer, int size, int offset, int length) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)`

This method transfers floats into this buffer from the given source array.

If there are more floats to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 860), then no floats are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

buffer The array from which floats are to be read.
size The size of the passed in buffer.
offset The offset within the array of the first float to be read.
length The number of floats to be read from the given array.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer
ReadOnlyBufferException (p. 2966) if this buffer is read-only
NullPointerException if the passed buffer is null.
IndexOutOfBoundsException if the preconditions of size, offset, or length are not met.

6.366.3.19 FloatBuffer& decaf::nio::FloatBuffer::put (std::vector< float > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source floats array into this buffer.

This is the same as calling put(&buffer[0], 0, buffer.size()).

Parameters

buffer The buffer whose contents are copied to this **FloatBuffer** (p. 1800)

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer

ReadOnlyBufferException (p. 2966) if this buffer is read-only

6.366.3.20 FloatBuffer& decaf::nio::FloatBuffer::put (FloatBuffer & *src*) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException)

This method transfers the floats remaining in the given source buffer into this buffer.

If there are more floats remaining in the source buffer than in this buffer, that is, if *src.remaining()* > **remaining()** (p. 860), then no floats are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies *n* = *src.remaining()* floats from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by *n*.

Parameters

src The buffer to take floats from an place in this one.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer for the remaining floats in the source buffer

IllegalArgumentException if the source buffer is this buffer.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

6.366.3.21 virtual FloatBuffer* decaf::nio::FloatBuffer::slice () const [pure virtual]

Creates a new **FloatBuffer** (p. 1800) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **FloatBuffer** (p.1800) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p.1800).

6.366.3.22 virtual std::string decaf::nio::FloatBuffer::toString () const [virtual]

Returns

a std::string describing this object

6.366.3.23 static FloatBuffer* decaf::nio::FloatBuffer::wrap (float * array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [static]

Wraps the passed buffer with a new **FloatBuffer** (p.1800).

The new buffer will be backed by the given float array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array The array that will back the new buffer.

size The size of the array that was passed in.

offset The offset of the subarray to be used.

length The length of the subarray to be used.

Returns

a new **FloatBuffer** (p.1800) that is backed by buffer, caller owns.

Exceptions

NullPointerException if the array pointer is NULL.

IndexOutOfBoundsException if the preconditions of size, offset, or length are not met.

6.366.3.24 static FloatBuffer* decaf::nio::FloatBuffer::wrap (std::vector< float > & buffer) [static]

Wraps the passed STL float Vector in a **FloatBuffer** (p.1800).

The new buffer will be backed by the given float array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

- buffer* - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new **FloatBuffer** (p. 1800) that is backed by *buffer*, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/FloatBuffer.h`

6.367 decaf::io::Flushable Class Reference

A **Flushable** (p. 1811) is a destination of data that can be flushed.

```
#include <src/main/decaf/io/Flushable.h>
```

Inheritance diagram for `decaf::io::Flushable`:

Public Member Functions

- virtual `~Flushable()`
- virtual void `flush()` throw (`decaf::io::IOException`)
Flushes this stream by writing any buffered output to the underlying stream.

6.367.1 Detailed Description

A **Flushable** (p. 1811) is a destination of data that can be flushed. The `flush` method is invoked to write any buffered output to the underlying stream.

Since

1.0

6.367.2 Constructor & Destructor Documentation

6.367.2.1 virtual `decaf::io::Flushable::~~Flushable()` [inline, virtual]

6.367.3 Member Function Documentation

6.367.3.1 virtual void `decaf::io::Flushable::flush()` throw (`decaf::io::IOException`) [pure virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Flushable.h`

6.368 activemq::commands::FlushCommand Class Reference

```
#include <src/main/activemq/commands/FlushCommand.h>
```

Inheritance diagram for `activemq::commands::FlushCommand`:

Public Member Functions

- **FlushCommand** ()
- virtual **~FlushCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **FlushCommand * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_FLUSHCOMMAND** = 15

6.368.1 Constructor & Destructor Documentation

6.368.1.1 `activemq::commands::FlushCommand::FlushCommand ()`

6.368.1.2 `virtual activemq::commands::FlushCommand::~~FlushCommand ()`
[virtual]

6.368.2 Member Function Documentation

6.368.2.1 `virtual FlushCommand* activemq::commands::FlushCommand::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.368.2.2 `virtual void activemq::commands::FlushCommand::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.368.2.3 `virtual bool activemq::commands::FlushCommand::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.368.2.4 `virtual unsigned char activemq::commands::FlushCommand::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1553) type copy.

Implements **activemq::commands::DataSet** (p. 1557).

6.368.2.5 **virtual std::string activemq::commands::FlushCommand::toString ()**
const [virtual]

Returns a string containing the information for this **DataSet** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

6.368.2.6 **virtual Pointer<Command> activemq::commands::FlushCommand::visit**
(activemq::state::CommandVisitor * *visitor*) throw (
exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.368.3 Field Documentation

6.368.3.1 **const unsigned char activemq::commands::FlushCommand::ID_ -**
FLUSHCOMMAND = 15 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**FlushCommand.h**

6.369 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1814).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller**:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.369.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1814). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.369.2 Constructor & Destructor Documentation

6.369.2.1 `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::FlushCommandM`
`() [inline]`

6.369.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::~~FlushCommand`
`() [inline, virtual]`

6.369.3 Member Function Documentation

6.369.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.369.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.369.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.369.3.4 virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 730).

6.369.3.5 virtual int activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 731).

6.369.3.6 virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

```
6.369.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h`

6.370 `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `FlushCommandMarshaller` (p. 1818).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller`:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.370.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1818). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.370.2 Constructor & Destructor Documentation

6.370.2.1 `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::FlushCommandM`
`() [inline]`

6.370.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::~~FlushCommand`
`() [inline, virtual]`

6.370.3 Member Function Documentation

6.370.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.370.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.370.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.370.3.4 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 737).

6.370.3.5 virtual int activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 738).

6.370.3.6 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

```
6.370.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h`

6.371 `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `FlushCommandMarshaller` (p. 1822).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller`:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.371.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1822). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.371.2 Constructor & Destructor Documentation

6.371.2.1 `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::FlushCommandM`
`() [inline]`

6.371.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::~~FlushCommand`
`() [inline, virtual]`

6.371.3 Member Function Documentation

6.371.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.371.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.371.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.371.3.4 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 703).

6.371.3.5 virtual int activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 704).

6.371.3.6 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.371.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h`

6.372 `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `FlushCommandMarshaller` (p. 1826).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller`:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.372.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1826). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.372.2 Constructor & Destructor Documentation

6.372.2.1 `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::FlushCommandM`
`() [inline]`

6.372.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::~~FlushCommand`
`() [inline, virtual]`

6.372.3 Member Function Documentation

6.372.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.372.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.372.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.372.3.4 virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 710).

6.372.3.5 virtual int activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 711).

6.372.3.6 virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.372.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h`

6.373 `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `FlushCommandMarshaller` (p. 1830).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller`:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.373.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1830). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.373.2 Constructor & Destructor Documentation

6.373.2.1 `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::FlushCommandM`
`() [inline]`

6.373.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::~~FlushCommand`
`() [inline, virtual]`

6.373.3 Member Function Documentation

6.373.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.373.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.373.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.373.3.4 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 717).

6.373.3.5 virtual int activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 718).

6.373.3.6 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

```
6.373.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h`

6.374 `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `FlushCommandMarshaller` (p. 1834).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller`:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.374.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1834). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.374.2 Constructor & Destructor Documentation

6.374.2.1 `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::FlushCommandMarshaller()` [inline]

6.374.2.2 `virtual activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::~~FlushCommandMarshaller()` [inline, virtual]

6.374.3 Member Function Documentation

6.374.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.374.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.374.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.374.3.4 virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 723).

6.374.3.5 virtual int activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 724).

6.374.3.6 virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 726).

```
6.374.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h`

6.375 decaf::util::logging::Formatter Class Reference

A **Formatter** (p. 1838) provides support for formatting LogRecords.

```
#include <src/main/decaf/util/logging/Formatter.h>
```

Inheritance diagram for decaf::util::logging::Formatter:

Public Member Functions

- virtual `~Formatter` ()
- virtual `std::string format` (const **LogRecord** &record) const =0
Format the given log record and return the formatted string.
- virtual `std::string formatMessage` (const **LogRecord** &record) const
Format the message string from a log record.
- virtual `std::string getHead` (const **Handler** *handler DECAF_UNUSED)
Return the header string for a set of formatted records.
- virtual `std::string getTail` (const **Handler** *handler DECAF_UNUSED)
Return the tail string for a set of formatted records.

6.375.1 Detailed Description

A **Formatter** (p. 1838) provides support for formatting LogRecords. Typically each logging **Handler** (p. 1852) will have a **Formatter** (p. 1838) associated with it. The **Formatter** (p. 1838) takes a **LogRecord** (p. 2261) and converts it to a string.

Some formatters (such as the **XMLFormatter** (p. 3793)) need to wrap head and tail strings around a set of formatted records. The `getHeader` and `getTail` methods can be used to obtain these strings.

6.375.2 Constructor & Destructor Documentation

6.375.2.1 virtual `decaf::util::logging::Formatter::~~Formatter` () [inline, virtual]

6.375.3 Member Function Documentation

6.375.3.1 virtual `std::string decaf::util::logging::Formatter::format` (const **LogRecord** & *record*) const [pure virtual]

Format the given log record and return the formatted string.

Parameters

record The Log Record to Format

Returns

the formatted record.

Implemented in `decaf::util::logging::SimpleFormatter` (p. 3279), and `decaf::util::logging::XMLFormatter` (p. 3794).

6.375.3.2 `virtual std::string decaf::util::logging::Formatter::formatMessage (const LogRecord & record) const` [virtual]

Format the message string from a log record.

Parameters

record The Log Record to Format

Returns

the formatted message

6.375.3.3 `virtual std::string decaf::util::logging::Formatter::getHead (const Handler *handler DECAF_UNUSED)` [inline, virtual]

Return the header string for a set of formatted records.

In the default implementation this method should return empty string.

Parameters

handler The target handler, can be NULL.

Returns

the head string.

6.375.3.4 `virtual std::string decaf::util::logging::Formatter::getTail (const Handler *handler DECAF_UNUSED)` [inline, virtual]

Return the tail string for a set of formatted records.

In the default implementation this method should return empty string

Parameters

handler the target handler, can be null

Returns

the tail string

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Formatter.h`

6.376 `decaf::util::concurrent::Future< V >` Class Template Reference

A **Future** (p. 1840) represents the result of an asynchronous computation.

`#include <src/main/decaf/util/concurrent/Future.h>`

Public Member Functions

- virtual **~Future** ()
- bool **cancel** (bool mayInterruptIfRunning)=0
Attempts to cancel execution of this task.
- bool **isCancelled** () const =0
Returns true if this task was canceled before it completed normally.
- bool **isDone** () const =0
Returns true if this task completed.
- V **get** ()=0 throw (CancellationException, InterruptedException, ExecutionException)
Waits if necessary for the computation to complete, and then retrieves its result.
- V **get** (long long timeout, **TimeUnit** unit)=0 throw (InterruptedException, ExecutionException, TimeoutException)
Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

6.376.1 Detailed Description

`template<typename V> class decaf::util::concurrent::Future< V >`

A **Future** (p.1840) represents the result of an asynchronous computation. Methods are provided to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation. The result can only be retrieved using method `get` when the computation has completed, blocking if necessary until it is ready. Cancellation is performed by the `cancel` method. Additional methods are provided to determine if the task completed normally or was canceled. Once a computation has completed, the computation cannot be canceled. If you would like to use a **Future** (p.1840) for the sake of cancellability but not provide a usable result, you can declare types of the form `Future<void*>` and return null as a result of the underlying task.

6.376.2 Constructor & Destructor Documentation

6.376.2.1 `template<typename V > virtual decaf::util::concurrent::Future< V >::~~Future () [inline, virtual]`

6.376.3 Member Function Documentation

6.376.3.1 `template<typename V > bool decaf::util::concurrent::Future< V >::cancel (bool mayInterruptIfRunning) [pure virtual]`

Attempts to cancel execution of this task.

This attempt will fail if the task has already completed, has already been canceled, or could not be canceled for some other reason. If successful, and this task has not started when `cancel` is called, this task should never run. If the task has already started, then the `mayInterruptIfRunning` parameter determines whether the thread executing this task should be interrupted in an attempt to stop the task.

After this method returns, subsequent calls to **isDone()** (p. 1843) will always return true. Subsequent calls to **isCancelled()** (p. 1843) will always return true if this method returned true.

Parameters

mayInterruptIfRunning - true if the thread executing this task should be interrupted; otherwise, in-progress tasks are allowed to complete.

Returns

false if the task could not be canceled, typically because it has already completed normally;
true otherwise

6.376.3.2 `template<typename V> V decaf::util::concurrent::Future< V >::get (long long timeout, TimeUnit unit) throw (InterruptedException, ExecutionException, TimeoutException) [pure virtual]`

Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

Parameters

timeout - the maximum time to wait

unit - the time unit of the timeout argument

Returns

the computed result

Exceptions

CancellationException (p. 1004) - if the computation was canceled

ExecutionException (p. 1746) - if the computation threw an exception

InterruptedException - if the current thread was interrupted while waiting

TimeoutException (p. 3541) - if the wait timed out

6.376.3.3 `template<typename V> V decaf::util::concurrent::Future< V >::get () throw (CancellationException, InterruptedException, ExecutionException) [pure virtual]`

Waits if necessary for the computation to complete, and then retrieves its result.

Returns

the computed result.

Exceptions

CancellationException (p. 1004) - if the computation was canceled

ExecutionException (p. 1746) - if the computation threw an exception

InterruptedException - if the current thread was interrupted while waiting

6.376.3.4 `template<typename V> bool decaf::util::concurrent::Future< V>::isCancelled () const [pure virtual]`

Returns true if this task was canceled before it completed normally.

Returns

true if this task was canceled before it completed

6.376.3.5 `template<typename V> bool decaf::util::concurrent::Future< V>::isDone () const [pure virtual]`

Returns true if this task completed.

Completion may be due to normal termination, an exception, or cancellation -- in all of these cases, this method will return true.

Returns

true if this task completed

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Future.h`

6.377 activemq::transport::correlator::FutureResponse Class Reference

A container that holds a response object.

```
#include <src/main/activemq/transport/correlator/FutureResponse.h>
```

Public Member Functions

- `FutureResponse ()`
- `virtual ~FutureResponse ()`
- `virtual const Pointer< Response > & getResponse () const`
Getters for the response property.
- `virtual Pointer< Response > & getResponse ()`
- `virtual const Pointer< Response > & getResponse (unsigned int timeout) const`
Getters for the response property.
- `virtual Pointer< Response > & getResponse (unsigned int timeout)`
- `virtual void setResponse (const Pointer< Response > &response)`
Setter for the response property.

6.377.1 Detailed Description

A container that holds a response object. Callers of the `getResponse` method will block until a response has been received unless they call the `getRepsonse` that takes a timeout.

6.377.2 Constructor & Destructor Documentation

6.377.2.1 `activemq::transport::correlator::FutureResponse::FutureResponse ()`
[inline]

6.377.2.2 `virtual`
`activemq::transport::correlator::FutureResponse::~~FutureResponse ()`
[inline, virtual]

6.377.3 Member Function Documentation

6.377.3.1 `virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse () const`
[inline, virtual]

Getters for the response property.

Infinite Wait.

Returns

the response object for the request

6.377.3.2 `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse ()`
[inline, virtual]

6.377.3.3 `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse (unsigned int timeout)` [inline, virtual]

6.377.3.4 `virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse (unsigned int timeout) const` [inline, virtual]

Getters for the response property.

Timed Wait.

Parameters

timeout - time to wait in milliseconds

Returns

the response object for the request

6.377.3.5 `virtual void activemq::transport::correlator::FutureResponse::setResponse (const Pointer< Response > & response)` [inline, virtual]

Setter for the response property.

Parameters

response the response object for the request.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/correlator/FutureResponse.h`

6.378 decaf::security::GeneralSecurityException Class Reference

```
#include <src/main/decaf/security/GeneralSecurityException.h>
```

Inheritance diagram for decaf::security::GeneralSecurityException:

Public Member Functions

- **GeneralSecurityException** () throw ()
Default Constructor.
- **GeneralSecurityException** (const decaf::lang::Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **GeneralSecurityException** (const GeneralSecurityException &ex) throw ()
Copy Constructor.
- **GeneralSecurityException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **GeneralSecurityException** (const std::exception *cause) throw ()
Constructor.
- **GeneralSecurityException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **GeneralSecurityException** * clone () const
Clones this exception.
- virtual ~**GeneralSecurityException** () throw ()

6.378.1 Constructor & Destructor Documentation

6.378.1.1 decaf::security::GeneralSecurityException::GeneralSecurityException () throw () [inline]

Default Constructor.

6.378.1.2 `decaf::security::GeneralSecurityException::GeneralSecurityException (const decaf::lang::Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.378.1.3 `decaf::security::GeneralSecurityException::GeneralSecurityException (const GeneralSecurityException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.378.1.4 `decaf::security::GeneralSecurityException::GeneralSecurityException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.378.1.5 `decaf::security::GeneralSecurityException::GeneralSecurityException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.378.1.6 `decaf::security::GeneralSecurityException::GeneralSecurityException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs
lineNumber line number where the exception occurred.
msg message to report
... list of primitives that are formatted into the message

6.378.1.7 virtual
decaf::security::GeneralSecurityException::~~GeneralSecurityException (
) throw () [inline, virtual]

6.378.2 Member Function Documentation

6.378.2.1 virtual GeneralSecurityException* de-
caf::security::GeneralSecurityException::clone () const
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::lang::Exception** (p. 1715).

Reimplemented in **decaf::security::cert::CertificateEncodingException**
(p. 1011), **decaf::security::cert::CertificateException** (p. 1013), **de-**
caf::security::cert::CertificateExpiredException (p. 1015), **de-**
caf::security::cert::CertificateNotYetValidException (p. 1017), **de-**
caf::security::cert::CertificateParsingException (p. 1019), **de-**
caf::security::InvalidKeyException (p. 1996), **decaf::security::KeyException**
(p. 2153), **decaf::security::KeyManagementException** (p. 2155),
decaf::security::NoSuchAlgorithmException (p. 2645), **de-**
caf::security::NoSuchProviderException (p. 2650), and **de-**
caf::security::SignatureException (p. 3278).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/GeneralSecurityException.h`

6.379 decaf::internal::util::GenericResource< T > Class Template Reference

A Generic **Resource** (p. 3072) wraps some type and will delete it when the **Resource** (p. 3072) itself is deleted.

```
#include <src/main/decaf/internal/util/GenericResource.h>
```

Inheritance diagram for decaf::internal::util::GenericResource< T >:

Public Member Functions

- **GenericResource** (T *value)
- virtual ~**GenericResource** ()
- T * **getManaged** () const
- void **setManaged** (T *value)

6.379.1 Detailed Description

`template<typename T> class decaf::internal::util::GenericResource< T >`

A Generic **Resource** (p. 3072) wraps some type and will delete it when the **Resource** (p. 3072) itself is deleted.

Since

1.0

6.379.2 Constructor & Destructor Documentation

6.379.2.1 `template<typename T> decaf::internal::util::GenericResource< T >::GenericResource (T * value) [inline, explicit]`

6.379.2.2 `template<typename T> virtual decaf::internal::util::GenericResource< T >::~~GenericResource () [inline, virtual]`

6.379.3 Member Function Documentation

6.379.3.1 `template<typename T> T* decaf::internal::util::GenericResource< T >::getManaged () const [inline]`

6.379.3.2 `template<typename T> void decaf::internal::util::GenericResource< T >::setManaged (T * value) [inline]`

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/GenericResource.h`

6.380 gz_header_s Struct Reference

`#include <src/main/decaf/internal/util/zip/zlib.h>`

Data Fields

- int **text**
- uLong **time**
- int **xflags**
- int **os**
- Bytef * **extra**
- uInt **extra_len**

- `uInt extra_max`
- `Bytef * name`
- `uInt name_max`
- `Bytef * comment`
- `uInt comm_max`
- `int hcrc`
- `int done`

6.380.1 Field Documentation

6.380.1.1 `uInt gz_header_s::comm_max`

6.380.1.2 `Bytef* gz_header_s::comment`

6.380.1.3 `int gz_header_s::done`

6.380.1.4 `Bytef* gz_header_s::extra`

6.380.1.5 `uInt gz_header_s::extra_len`

6.380.1.6 `uInt gz_header_s::extra_max`

6.380.1.7 `int gz_header_s::hcrc`

6.380.1.8 `Bytef* gz_header_s::name`

6.380.1.9 `uInt gz_header_s::name_max`

6.380.1.10 `int gz_header_s::os`

6.380.1.11 `int gz_header_s::text`

6.380.1.12 `uLong gz_header_s::time`

6.380.1.13 `int gz_header_s::xflags`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/zlib.h`

6.381 gz_state Struct Reference

```
#include <src/main/decaf/internal/util/zip/gzguts.h>
```

Data Fields

- `int mode`
- `int fd`
- `char * path`
- `z_off64_t pos`

- unsigned **size**
- unsigned **want**
- unsigned char * **in**
- unsigned char * **out**
- unsigned char * **next**
- unsigned **have**
- int **eof**
- z_off64_t **start**
- z_off64_t **raw**
- int **how**
- int **direct**
- int **level**
- int **strategy**
- z_off64_t **skip**
- int **seek**
- int **err**
- char * **msg**
- z_stream **strm**

6.381.1 Field Documentation

- 6.381.1.1 int gz_state::direct
- 6.381.1.2 int gz_state::eof
- 6.381.1.3 int gz_state::err
- 6.381.1.4 int gz_state::fd
- 6.381.1.5 unsigned gz_state::have
- 6.381.1.6 int gz_state::how
- 6.381.1.7 unsigned char* gz_state::in
- 6.381.1.8 int gz_state::level
- 6.381.1.9 int gz_state::mode
- 6.381.1.10 char* gz_state::msg
- 6.381.1.11 unsigned char* gz_state::next
- 6.381.1.12 unsigned char* gz_state::out
- 6.381.1.13 char* gz_state::path
- 6.381.1.14 z_off64_t gz_state::pos
- 6.381.1.15 z_off64_t gz_state::raw
- 6.381.1.16 int gz_state::seek
- 6.381.1.17 unsigned gz_state::size
- 6.381.1.18 z_off64_t gz_state::skip
- 6.381.1.19 z_off64_t gz_state::start
- 6.381.1.20 int gz_state::strategy
- 6.381.1.21 z_stream gz_state::strm
- 6.381.1.22 unsigned gz_state::want

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**gzguts.h**

6.382 decaf::util::logging::Handler Class Reference

A **Handler** (p. 1852) object takes log messages from a **Logger** (p. 2237) and exports them.

```
#include <src/main/decaf/util/logging/Handler.h>
```

Inheritance diagram for decaf::util::logging::Handler:

Public Member Functions

- **Handler** ()
- virtual **~Handler** ()
- virtual void **flush** ()=0
Flush the Handler's output, clears any buffers.
- virtual void **publish** (const **LogRecord** &record)=0
*Publish the Log Record to this **Handler** (p. 1852).*
- virtual bool **isLoggable** (const **LogRecord** &record) const
*Check if this **Handler** (p. 1852) would actually log a given **LogRecord** (p. 2261).*
- virtual void **setFilter** (**Filter** *filter)
*Sets the **Filter** (p. 1770) that this **Handler** (p. 1852) uses to filter Log Records.*
- virtual **Filter** * **getFilter** ()
*Gets the **Filter** (p. 1770) that this **Handler** (p. 1852) uses to filter Log Records.*
- virtual void **setLevel** (const **Level** &value)
*Set (p. 3220) the log level specifying which message levels will be logged by this **Handler** (p. 1852).*
- virtual **Level** **getLevel** ()
*Get the log level specifying which message levels will be logged by this **Handler** (p. 1852).*
- virtual void **setFormatter** (**Formatter** *formatter)
*Sets the **Formatter** (p. 1838) used by this **Handler** (p. 1852).*
- virtual **Formatter** * **getFormatter** ()
*Gets the **Formatter** (p. 1838) used by this **Handler** (p. 1852).*
- virtual void **setErrorManager** (**ErrorManager** *errorManager)
*Sets the **Formatter** (p. 1838) used by this **Handler** (p. 1852).*
- virtual **ErrorManager** * **getErrorManager** ()
*Gets the **ErrorManager** (p. 1710) used by this **Handler** (p. 1852).*

Protected Member Functions

- void **reportError** (const std::string &message, **decaf::lang::Exception** *ex, int code)
*Protected convenience method to report an error to this Handler's **ErrorManager** (p. 1710).*

6.382.1 Detailed Description

A **Handler** (p. 1852) object takes log messages from a **Logger** (p. 2237) and exports them. It might for example, write them to a console or write them to a file, or send them to a network logging service, or forward them to an OS log, or whatever.

A **Handler** (p. 1852) can be disabled by doing a **setLevel(Level.OFF** (p. 2190)) and can be re-enabled by doing a **setLevel** with an appropriate level.

Handler (p. 1852) classes typically use **LogManager** (p. 2254) properties to set default values for the Handler's **Filter** (p. 1770), **Formatter** (p. 1838), and **Level** (p. 2185). See the specific documentation for each concrete **Handler** (p. 1852) class.

6.382.2 Constructor & Destructor Documentation

6.382.2.1 **decaf::util::logging::Handler::Handler** ()

6.382.2.2 **virtual decaf::util::logging::Handler::~~Handler** () [virtual]

6.382.3 Member Function Documentation

6.382.3.1 **virtual void decaf::util::logging::Handler::flush** () [pure virtual]

Flush the Handler's output, clears any buffers.

Implemented in **decaf::util::logging::StreamHandler** (p. 3414).

6.382.3.2 **virtual ErrorManager* decaf::util::logging::Handler::getErrorManager** () [inline, virtual]

Gets the **ErrorManager** (p. 1710) used by this **Handler** (p. 1852).

Returns

ErrorManager (p. 1710) derived pointer or NULL.

6.382.3.3 **virtual Filter* decaf::util::logging::Handler::getFilter** () [inline, virtual]

Gets the **Filter** (p. 1770) that this **Handler** (p. 1852) uses to filter Log Records.

Returns

Filter (p. 1770) derived instance

6.382.3.4 `virtual Formatter* decaf::util::logging::Handler::getFormatter ()`
`[inline, virtual]`

Gets the `Formatter` (p. 1838) used by this `Handler` (p. 1852).

Returns

`Filter` (p. 1770) derived instance

6.382.3.5 `virtual Level decaf::util::logging::Handler::getLevel ()` `[inline, virtual]`

Get the log level specifying which message levels will be logged by this `Handler` (p. 1852).

Returns

`Level` (p. 2185) enumeration value

6.382.3.6 `virtual bool decaf::util::logging::Handler::isLoggable (const LogRecord & record) const` `[virtual]`

Check if this `Handler` (p. 1852) would actually log a given `LogRecord` (p. 2261).

This method checks if the `LogRecord` (p. 2261) has an appropriate `Level` (p. 2185) and whether it satisfies any `Filter` (p. 1770). It also may make other `Handler` (p. 1852) specific checks that might prevent a handler from logging the `LogRecord` (p. 2261).

Parameters

record `LogRecord` (p. 2261) to check

Reimplemented in `decaf::util::logging::StreamHandler` (p. 3414).

6.382.3.7 `virtual void decaf::util::logging::Handler::publish (const LogRecord & record)` `[pure virtual]`

Publish the Log Record to this `Handler` (p. 1852).

Parameters

record The Log Record to Publish

Implemented in `decaf::util::logging::ConsoleHandler` (p. 1302), and `decaf::util::logging::StreamHandler` (p. 3415).

6.382.3.8 `void decaf::util::logging::Handler::reportError (const std::string & message, decaf::lang::Exception * ex, int code)` `[protected]`

Protected convenience method to report an error to this Handler's `ErrorManager` (p. 1710).

Parameters

message - a descriptive string (may be empty)

ex - an exception (may be NULL)

code - an error code defined in **ErrorManager** (p. 1710)

6.382.3.9 virtual void decaf::util::logging::Handler::setErrorHandler (**ErrorManager** * *errorManager*) [virtual]

Sets the **Formatter** (p. 1838) used by this **Handler** (p. 1852).

The **ErrorManager**'s "error" method will be invoked if any errors occur while using this **Handler** (p. 1852).

Parameters

errorManager **ErrorManager** (p. 1710) derived instance

6.382.3.10 virtual void decaf::util::logging::Handler::setFilter (**Filter** * *filter*) [inline, virtual]

Sets the **Filter** (p. 1770) that this **Handler** (p. 1852) uses to filter Log Records.

For each call of publish the **Handler** (p. 1852) will call this **Filter** (p. 1770) (if it is non-null) to check if the **LogRecord** (p. 2261) should be published or discarded.

Parameters

filter **Filter** (p. 1770) derived instance

6.382.3.11 virtual void decaf::util::logging::Handler::setFormatter (**Formatter** * *formatter*) [virtual]

Sets the **Formatter** (p. 1838) used by this **Handler** (p. 1852).

Some Handlers may not use Formatters, in which case the **Formatter** (p. 1838) will be remembered, but not used.

Parameters

formatter **Filter** (p. 1770) derived instance

6.382.3.12 virtual void decaf::util::logging::Handler::setLevel (const **Level** & *value*) [inline, virtual]

Set (p. 3220) the log level specifying which message levels will be logged by this **Handler** (p. 1852).

The intention is to allow developers to turn on voluminous logging, but to limit the messages that are sent to certain Handlers.

Parameters

value **Level** (p. 2185) enumeration value

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Handler.h**

6.383 decaf::internal::util::HexStringParser Class Reference

```
#include <src/main/decaf/internal/util/HexStringParser.h>
```

Public Member Functions

- **HexStringParser** (int exponentWidth, int mantissaWidth)
Create a new HexParser.
- virtual **~HexStringParser** ()
- long long **parse** (const std::string &hexString)
Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.

Static Public Member Functions

- static double **parseDouble** (const std::string &hexString)
- static float **parseFloat** (const std::string &hexString)

6.383.1 Constructor & Destructor Documentation

6.383.1.1 decaf::internal::util::HexStringParser::HexStringParser (int exponentWidth, int mantissaWidth)

Create a new HexParser.

Parameters

exponentWidth - Width of the exponent for the type to parse
mantissaWidth - Width of the mantissa for the type to parse

6.383.1.2 virtual decaf::internal::util::HexStringParser::~~HexStringParser () [inline, virtual]

6.383.2 Member Function Documentation

6.383.2.1 long long decaf::internal::util::HexStringParser::parse (const std::string & hexString)

Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.

Parameters

hexString - string to parse

Returns

the bits parsed from the string

6.383.2.2 static double decaf::internal::util::HexStringParser::parseDouble (const std::string & *hexString*) [static]

6.383.2.3 static float decaf::internal::util::HexStringParser::parseFloat (const std::string & *hexString*) [static]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**HexStringParser.h**

6.384 activemq::wireformat::openwire::utils::HexTable Class Reference

The **HexTable** (p. 1857) class maps hexadecimal strings to the value of an index into the table, i.e.

```
#include <src/main/activemq/wireformat/openwire/utils/HexTable.h>
```

Public Member Functions

- **HexTable** ()
- virtual ~**HexTable** ()
- virtual const std::string & **operator[]** (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.

- virtual const std::string & **operator[]** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual std::size_t **size** () const

Returns the max size of this Table.

6.384.1 Detailed Description

The **HexTable** (p. 1857) class maps hexadecimal strings to the value of an index into the table, i.e. the class will return "FF" for the index 255 in the table.

6.384.2 Constructor & Destructor Documentation

6.384.2.1 `activemq::wireformat::openwire::utils::HexTable::HexTable ()`

6.384.2.2 `virtual activemq::wireformat::openwire::utils::HexTable::~~HexTable ()`
[inline, virtual]

6.384.3 Member Function Documentation

6.384.3.1 `virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[] (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.

Parameters

index The index of the value in the table to fetch.

Returns

string containing the hex value if the index

Exceptions

IndexOutOfBoundsException

6.384.3.2 `virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[] (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

6.384.3.3 `virtual std::size_t activemq::wireformat::openwire::utils::HexTable::size () const` [inline, virtual]

Returns the max size of this Table.

Returns

an integer size value

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/HexTable.h`

6.385 decaf::net::HttpRetryException Class Reference

```
#include <src/main/decaf/net/HttpRetryException.h>
```

Inheritance diagram for decaf::net::HttpRetryException:

Public Member Functions

- **HttpRetryException** () throw ()
Default Constructor.
- **HttpRetryException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **HttpRetryException** (const **HttpRetryException** &ex) throw ()
Copy Constructor.
- **HttpRetryException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **HttpRetryException** (const std::exception *cause) throw ()
Constructor.
- **HttpRetryException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **HttpRetryException** * clone () const
Clones this exception.
- virtual ~**HttpRetryException** () throw ()

6.385.1 Constructor & Destructor Documentation

6.385.1.1 decaf::net::HttpRetryException::HttpRetryException () throw () [inline]

Default Constructor.

6.385.1.2 decaf::net::HttpRetryException::HttpRetryException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.385.1.3 decaf::net::HttpRetryException::HttpRetryException (const HttpRetryException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.385.1.4 `decaf::net::HttpRetryException::HttpRetryException (const char *
file, const int lineNumber, const std::exception * cause, const char *
msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.385.1.5 `decaf::net::HttpRetryException::HttpRetryException (const
std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.385.1.6 `decaf::net::HttpRetryException::HttpRetryException (const char * file,
const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.385.1.7 `virtual decaf::net::HttpRetryException::~~HttpRetryException ()
throw () [inline, virtual]`

6.385.2 Member Function Documentation

6.385.2.1 `virtual HttpRetryException* decaf::net::HttpRetryException::clone ()
const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2005).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/HttpRetryException.h`

6.386 activemq::util::IdGenerator Class Reference

```
#include <src/main/activemq/util/IdGenerator.h>
```

Data Structures

- class **StaticData**

Public Member Functions

- **IdGenerator** ()
- **IdGenerator** (const std::string &prefix)
- virtual **~IdGenerator** ()
- std::string **generateId** () const

Static Public Member Functions

- static std::string **getHostname** ()
Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.
- static std::string **getSeedFromId** (const std::string &id)
Gets the seed value from a Generated Id, the count portion is removed.
- static long long **getSequenceFromId** (const std::string &id)
Gets the count value from a Generated Id, the seed portion is removed.

- static int **compare** (const std::string &id1, const std::string &id2)
Compares two generated id values.

6.386.1 Constructor & Destructor Documentation

6.386.1.1 `activemq::util::IdGenerator::IdGenerator ()`

6.386.1.2 `activemq::util::IdGenerator::IdGenerator (const std::string & prefix)`

6.386.1.3 `virtual activemq::util::IdGenerator::~~IdGenerator ()` [virtual]

6.386.2 Member Function Documentation

6.386.2.1 `static int activemq::util::IdGenerator::compare (const std::string & id1, const std::string & id2)` [static]

Compares two generated id values.

Parameters

- id1* The first id to compare, or left hand side.
- id2* The second id to compare, or right hand side.

Returns

zero if ids are equal or positive if id1 > id2...

6.386.2.2 `std::string activemq::util::IdGenerator::generateId ()` const

Returns

a newly generated unique id.

6.386.2.3 `static std::string activemq::util::IdGenerator::getHostname ()` [static]

Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.

Returns

the previously retrieved host name.

6.386.2.4 `static std::string activemq::util::IdGenerator::getSeedFromId (const std::string & id)` [static]

Gets the seed value from a Generated Id, the count portion is removed.

Returns

the seed portion of the Id, minus the count value.

6.386.2.5 static long long activemq::util::IdGenerator::getSequenceFromId (const std::string & id) [static]

Gets the count value from a Generated Id, the seed portion is removed.

Returns

the sequence count portion of the id, minus the seed value.

The documentation for this class was generated from the following file:

- src/main/activemq/util/IdGenerator.h

6.387 decaf::lang::exceptions::IllegalArgumentException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalArgumentException.h>
```

Inheritance diagram for decaf::lang::exceptions::IllegalArgumentException:

Public Member Functions

- **IllegalArgumentException** () throw ()
Default Constructor.
- **IllegalArgumentException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1712).*
- **IllegalArgumentException** (const **IllegalArgumentException** &ex) throw ()
Copy Constructor.
- **IllegalArgumentException** (const std::exception *cause) throw ()
Constructor.
- **IllegalArgumentException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalArgumentException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **IllegalArgumentException** * clone () const
Clones this exception.
- virtual ~**IllegalArgumentException** () throw ()

6.387.1 Constructor & Destructor Documentation

6.387.1.1 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException
() throw () [inline]`

Default Constructor.

6.387.1.2 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException
(const Exception & ex) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.387.1.3 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException
(const IllegalArgumentException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.387.1.4 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException
(const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause **Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.387.1.5 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException
(const char * file, const int lineNumber, const char * msg, ...)
throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.387.1.6 decaf::lang::exceptions::IllegalArgumentOutOfRangeException::IllegalArgumentOutOfRangeException
 (const char * *file*, const int *lineNumber*, const std::exception *
cause, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.387.1.7 virtual
 decaf::lang::exceptions::IllegalArgumentOutOfRangeException::~~IllegalArgumentOutOfRangeException
 () throw () [inline, virtual]

6.387.2 Member Function Documentation

6.387.2.1 virtual IllegalArgumentOutOfRangeException* de-
 caf::lang::exceptions::IllegalArgumentOutOfRangeException::clone (
) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/IllegalArgumentOutOfRangeException.h

6.388 decaf::lang::exceptions::IllegalMonitorStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalMonitorStateException.h>
```

Inheritance diagram for decaf::lang::exceptions::IllegalMonitorStateException:

Public Member Functions

- **IllegalMonitorStateException** () throw ()
Default Constructor.
- **IllegalMonitorStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1712).*
- **IllegalMonitorStateException** (const **IllegalMonitorStateException** &ex) throw ()
Copy Constructor.
- **IllegalMonitorStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalMonitorStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalMonitorStateException** (const std::exception *cause) throw ()
Constructor.
- virtual **IllegalMonitorStateException** * clone () const
Clones this exception.
- virtual ~**IllegalMonitorStateException** () throw ()

6.388.1 Constructor & Destructor Documentation

6.388.1.1 `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException () throw () [inline]`

Default Constructor.

6.388.1.2 `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.388.1.3 `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const IllegalMonitorStateException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.388.1.4 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException
 (const char * *file*, const int *lineNumber*, const char * *msg*, ...)
 throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.388.1.5 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException
 (const char * *file*, const int *lineNumber*, const std::exception *
cause, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.388.1.6 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException
 (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause **Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.388.1.7 virtual
 decaf::lang::exceptions::IllegalMonitorStateException::~~IllegalMonitorStateException
 () throw () [inline, virtual]

6.388.2 Member Function Documentation

6.388.2.1 virtual IllegalMonitorStateException* de-
 caf::lang::exceptions::IllegalMonitorStateException::clone (
) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalMonitorStateException.h`

6.389 cms::IllegalStateException Class Reference

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

```
#include <src/main/cms/IllegalStateException.h>
```

Inheritance diagram for cms::IllegalStateException:

Public Member Functions

- **IllegalStateException** () throw ()
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()
- **IllegalStateException** (const std::string &message, const std::exception *cause) throw ()
- **IllegalStateException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**IllegalStateException** () throw ()

6.389.1 Detailed Description

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation. For example, this exception must be thrown if **Session.commit** (p. 3152) is called on a non-transacted session.

Since

1.3

6.389.2 Constructor & Destructor Documentation

- 6.389.2.1 `cms::IllegalStateException::IllegalStateException () throw ()`
- 6.389.2.2 `cms::IllegalStateException::IllegalStateException (const
IllegalStateException & ex) throw ()`
- 6.389.2.3 `cms::IllegalStateException::IllegalStateException (const std::string &
message, const std::exception * cause) throw ()`
- 6.389.2.4 `cms::IllegalStateException::IllegalStateException (const std::string &
message, const std::exception * cause, const std::vector< std::pair<
std::string, int > > & stackTrace) throw ()`
- 6.389.2.5 `virtual cms::IllegalStateException::~~IllegalStateException () throw ()
[virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/IllegalStateException.h`

6.390 decaf::lang::exceptions::IllegalStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalStateException.h>
```

Inheritance diagram for decaf::lang::exceptions::IllegalStateException:

Public Member Functions

- `IllegalStateException () throw ()`
Default Constructor.
- `IllegalStateException (const Exception &ex) throw ()`
*Conversion Constructor from some other **Exception** (p. 1712).*
- `IllegalStateException (const IllegalStateException &ex) throw ()`
Copy Constructor.
- `IllegalStateException (const char *file, const int lineNumber, const char *msg,...) throw
()`
Constructor - Initializes the file name and line number where this message occurred.
- `IllegalStateException (const char *file, const int lineNumber, const std::exception *cause,
const char *msg,...) throw ()`
Constructor - Initializes the file name and line number where this message occurred.
- `IllegalStateException (const std::exception *cause) throw ()`

Constructor.

- virtual **IllegalStateException** * **clone** () const

Clones this exception.

- virtual ~**IllegalStateException** () throw ()

6.390.1 Constructor & Destructor Documentation

6.390.1.1 `decaf::lang::exceptions::IllegalStateException::IllegalStateException ()
throw () [inline]`

Default Constructor.

6.390.1.2 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (
const Exception & ex) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.390.1.3 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (
const IllegalStateException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.390.1.4 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (
const char * file, const int lineNumber, const char * msg, ...)
throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.390.1.5 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.390.1.6 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause **Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.390.1.7 `virtual decaf::lang::exceptions::IllegalStateException::~~IllegalStateException () throw () [inline, virtual]`

6.390.2 Member Function Documentation

6.390.2.1 `virtual IllegalStateException* decaf::lang::exceptions::IllegalStateException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

Reimplemented in **decaf::nio::InvalidMarkException** (p. 1999).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalStateException.h`

6.391 decaf::lang::exceptions::IllegalThreadStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalThreadStateException.h>
```

Inheritance diagram for decaf::lang::exceptions::IllegalThreadStateException:

Public Member Functions

- **IllegalThreadStateException** () throw ()
Default Constructor.
- **IllegalThreadStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1712).*
- **IllegalThreadStateException** (const **IllegalThreadStateException** &ex) throw ()
Copy Constructor.
- **IllegalThreadStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalThreadStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalThreadStateException** (const std::exception *cause) throw ()
Constructor.
- virtual **IllegalThreadStateException** * clone () const
Clones this exception.
- virtual ~**IllegalThreadStateException** () throw ()

6.391.1 Constructor & Destructor Documentation

6.391.1.1 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException () throw () [inline]

Default Constructor.

6.391.1.2 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.391.1.3 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException
 (const IllegalThreadStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.391.1.4 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException
 (const char * *file*, const int *lineNumber*, const char * *msg*, ...)
 throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.391.1.5 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException
 (const char * *file*, const int *lineNumber*, const std::exception *
cause, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.391.1.6 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException
 (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause **Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.391.1.7 `virtual`
`decaf::lang::exceptions::IllegalThreadStateException::~~IllegalThreadStateException`
`() throw () [inline, virtual]`

6.391.2 Member Function Documentation

6.391.2.1 `virtual IllegalThreadStateException* de-`
`caf::lang::exceptions::IllegalThreadStateException::clone (`
`) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalThreadStateException.h`

6.392 activemq::transport::inactivity::InactivityMonitor Class Reference

```
#include <src/main/activemq/transport/inactivity/InactivityMonitor.h>
```

Inheritance diagram for `activemq::transport::inactivity::InactivityMonitor`:

Public Member Functions

- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **decaf::util::Properties** &properties, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- virtual **~InactivityMonitor** ()
- virtual void **close** () throw (**decaf::io::IOException**)
Stops the polling thread and closes the streams.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)

Sends a one-way command.

- bool **isKeepAliveResponseRequired** () const
- void **setKeepAliveResponseRequired** (bool value)
- long long **getReadCheckTime** () const
- void **setReadCheckTime** (long long value)
- long long **getWriteCheckTime** () const
- void **setWriteCheckTime** (long long value)
- long long **getInitialDelayTime** () const
- void **setInitialDelayTime** (long long value) const

Friends

- class **ReadChecker**
- class **AsyncSignalReadErrorTask**
- class **WriteChecker**
- class **AsyncWriteTask**

6.392.1 Constructor & Destructor Documentation

6.392.1.1 `activemq::transport::inactivity::InactivityMonitor::InactivityMonitor (const Pointer< Transport > & next, const Pointer< wireformat::WireFormat > & wireFormat)`

6.392.1.2 `activemq::transport::inactivity::InactivityMonitor::InactivityMonitor (const Pointer< Transport > & next, const decaf::util::Properties & properties, const Pointer< wireformat::WireFormat > & wireFormat)`

6.392.1.3 `virtual
activemq::transport::inactivity::InactivityMonitor::~InactivityMonitor () [virtual]`

6.392.2 Member Function Documentation

6.392.2.1 `virtual void activemq::transport::inactivity::InactivityMonitor::close ()
throw (decaf::io::IOException) [virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if an error occurs while closing the **Transport** (p. 3629).

Reimplemented from **activemq::transport::TransportFilter** (p. 3638).

- 6.392.2.2** `long long activemq::transport::inactivity::InactivityMonitor::getInitialDelayTime () const`
- 6.392.2.3** `long long activemq::transport::inactivity::InactivityMonitor::getReadCheckTime () const`
- 6.392.2.4** `long long activemq::transport::inactivity::InactivityMonitor::getWriteCheckTime () const`
- 6.392.2.5** `bool activemq::transport::inactivity::InactivityMonitor::isKeepAliveResponseRequired () const`
- 6.392.2.6** `virtual void activemq::transport::inactivity::InactivityMonitor::onCommand (const Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

Parameters

command - the received command object.

Reimplemented from `activemq::transport::TransportFilter` (p. 3640).

- 6.392.2.7** `virtual void activemq::transport::inactivity::InactivityMonitor::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from `activemq::transport::TransportFilter` (p. 3640).

- 6.392.2.8** `virtual void activemq::transport::inactivity::InactivityMonitor::onException (const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters

ex The exception to handle.

Reimplemented from `activemq::transport::TransportFilter` (p. 3641).

- 6.392.2.9** `void activemq::transport::inactivity::InactivityMonitor::setInitialDelayTime (long long value) const`
- 6.392.2.10** `void activemq::transport::inactivity::InactivityMonitor::setKeepAliveResponseRequired (bool value)`
- 6.392.2.11** `void activemq::transport::inactivity::InactivityMonitor::setReadCheckTime (long long value)`
- 6.392.2.12** `void activemq::transport::inactivity::InactivityMonitor::setWriteCheckTime (long long value)`

6.392.3 Friends And Related Function Documentation

- 6.392.3.1** `friend class AsyncSignalReadErrorTask [friend]`
- 6.392.3.2** `friend class AsyncWriteTask [friend]`
- 6.392.3.3** `friend class ReadChecker [friend]`
- 6.392.3.4** `friend class WriteChecker [friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/inactivity/InactivityMonitor.h`

6.393 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference

```
#include <src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Inheritance diagram for `decaf::lang::exceptions::IndexOutOfBoundsException`:

Public Member Functions

- `IndexOutOfBoundsException () throw ()`
Default Constructor.
- `IndexOutOfBoundsException (const Exception &ex) throw ()`

*Conversion Constructor from some other **Exception** (p. 1712).*

- **IndexOutOfBoundsException** (const **IndexOutOfBoundsException** &ex) throw ()
Copy Constructor.
- **IndexOutOfBoundsException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IndexOutOfBoundsException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IndexOutOfBoundsException** (const std::exception *cause) throw ()
Constructor.
- virtual **IndexOutOfBoundsException** * clone () const
Clones this exception.
- virtual ~**IndexOutOfBoundsException** () throw ()

6.393.1 Constructor & Destructor Documentation

6.393.1.1 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException () throw () [inline]

Default Constructor.

6.393.1.2 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.393.1.3 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const IndexOutOfBoundsException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.393.1.4 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException
 (const char * *file*, const int *lineNumber*, const char * *msg*, ...)
 throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.393.1.5 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException
 (const char * *file*, const int *lineNumber*, const std::exception *
cause, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.393.1.6 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException
 (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause **Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.393.1.7 virtual
 decaf::lang::exceptions::IndexOutOfBoundsException::~~IndexOutOfBoundsException
 () throw () [inline, virtual]

6.393.2 Member Function Documentation

6.393.2.1 virtual IndexOutOfBoundsException* de-
 caf::lang::exceptions::IndexOutOfBoundsException::clone (
) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h`

6.394 decaf::net::Inet4Address Class Reference

```
#include <src/main/decaf/net/Inet4Address.h>
```

Inheritance diagram for decaf::net::Inet4Address:

Public Member Functions

- virtual `~Inet4Address ()`
- virtual bool `isAnyLocalAddress () const`
*Check if this **InetAddress** (p. 1884) is a valid wildcard address.*
- virtual bool `isLoopbackAddress () const`
*Check if this **InetAddress** (p. 1884) is a valid loopback address.*
- virtual bool `isMulticastAddress () const`
*Check if this **InetAddress** (p. 1884) is a valid Multicast address.*
- virtual bool `isLinkLocalAddress () const`
*Check if this **InetAddress** (p. 1884) is a valid link local address.*
- virtual bool `isSiteLocalAddress () const`
*Check if this **InetAddress** (p. 1884) is a valid site local address.*
- virtual bool `isMCGlobal () const`
*Check if this **InetAddress** (p. 1884) is Multicast and has Global scope.*
- virtual bool `isMCNodeLocal () const`
*Check if this **InetAddress** (p. 1884) is Multicast and has Node Local scope.*
- virtual bool `isMCLinkLocal () const`
*Check if this **InetAddress** (p. 1884) is Multicast and has Link Local scope.*
- virtual bool `isMCSiteLocal () const`
*Check if this **InetAddress** (p. 1884) is Multicast and has Site Local scope.*

- virtual bool **isMCOrgLocal** () const

*Check if this **InetAddress** (p. 1884) is Multicast and has Organization scope.*

Protected Member Functions

- **Inet4Address** ()
- **Inet4Address** (const unsigned char *ipAddress, int numBytes)
- **Inet4Address** (const std::string &hostname, const unsigned char *ipAddress, int numBytes)

Friends

- class **InetAddress**

6.394.1 Constructor & Destructor Documentation

6.394.1.1 decaf::net::Inet4Address::Inet4Address () [protected]

6.394.1.2 decaf::net::Inet4Address::Inet4Address (const unsigned char * *ipAddress*, int *numBytes*) [protected]

6.394.1.3 decaf::net::Inet4Address::Inet4Address (const std::string & *hostname*, const unsigned char * *ipAddress*, int *numBytes*) [protected]

6.394.1.4 virtual decaf::net::Inet4Address::~~Inet4Address () [virtual]

6.394.2 Member Function Documentation

6.394.2.1 virtual bool decaf::net::Inet4Address::isAnyLocalAddress () const [virtual]

Check if this **InetAddress** (p. 1884) is a valid wildcard address.

Returns

true if the address is a wildcard address.

Reimplemented from **decaf::net::InetAddress** (p. 1889).

6.394.2.2 virtual bool decaf::net::Inet4Address::isLinkLocalAddress () const [virtual]

Check if this **InetAddress** (p. 1884) is a valid link local address.

Returns

true if the address is a link local address.

Reimplemented from **decaf::net::InetAddress** (p. 1889).

6.394.2.3 `virtual bool decaf::net::Inet4Address::isLoopbackAddress () const`
[virtual]

Check if this **InetAddress** (p. 1884) is a valid loopback address.

Returns

true if the address is a loopback address.

Reimplemented from **decaf::net::InetAddress** (p. 1889).

6.394.2.4 `virtual bool decaf::net::Inet4Address::isMCGlobal () const` [virtual]

Check if this **InetAddress** (p. 1884) is Multicast and has Global scope.

Returns

true if the address is Multicast and has Global scope.

Reimplemented from **decaf::net::InetAddress** (p. 1889).

6.394.2.5 `virtual bool decaf::net::Inet4Address::isMCLinkLocal () const`
[virtual]

Check if this **InetAddress** (p. 1884) is Multicast and has Link Local scope.

Returns

true if the address is Multicast and has Link Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1889).

6.394.2.6 `virtual bool decaf::net::Inet4Address::isMCNodeLocal () const`
[virtual]

Check if this **InetAddress** (p. 1884) is Multicast and has Node Local scope.

Returns

true if the address is Multicast and has Node Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1890).

6.394.2.7 `virtual bool decaf::net::Inet4Address::isMCOrgLocal () const`
[virtual]

Check if this **InetAddress** (p. 1884) is Multicast and has Organization scope.

Returns

true if the address is Multicast and has Organization scope.

Reimplemented from **decaf::net::InetAddress** (p. 1890).

6.394.2.8 `virtual bool decaf::net::Inet4Address::isMCSiteLocal () const`
[virtual]

Check if this **InetAddress** (p. 1884) is Multicast and has Site Local scope.

Returns

true if the address is Multicast and has Site Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1890).

6.394.2.9 `virtual bool decaf::net::Inet4Address::isMulticastAddress () const`
[virtual]

Check if this **InetAddress** (p. 1884) is a valid Multicast address.

Returns

true if the address is a Multicast address.

Reimplemented from **decaf::net::InetAddress** (p. 1890).

6.394.2.10 `virtual bool decaf::net::Inet4Address::isSiteLocalAddress () const`
[virtual]

Check if this **InetAddress** (p. 1884) is a valid site local address.

Returns

true if the address is a site local address.

Reimplemented from **decaf::net::InetAddress** (p. 1890).

6.394.3 Friends And Related Function Documentation

6.394.3.1 `friend class InetAddress` [friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Inet4Address.h`

6.395 decaf::net::Inet6Address Class Reference

```
#include <src/main/decaf/net/Inet6Address.h>
```

Inheritance diagram for **decaf::net::Inet6Address**:

Public Member Functions

- `virtual ~Inet6Address ()`

Protected Member Functions

- **Inet6Address** ()
- **Inet6Address** (const unsigned char *ipAddress, int numBytes)
- **Inet6Address** (const std::string &hostname, const unsigned char *ipAddress, int numBytes)

Friends

- class **InetAddress**

6.395.1 Constructor & Destructor Documentation

- 6.395.1.1 **decaf::net::Inet6Address::Inet6Address** () [protected]
- 6.395.1.2 **decaf::net::Inet6Address::Inet6Address** (const unsigned char * *ipAddress*, int *numBytes*) [protected]
- 6.395.1.3 **decaf::net::Inet6Address::Inet6Address** (const std::string & *hostname*, const unsigned char * *ipAddress*, int *numBytes*) [protected]
- 6.395.1.4 **virtual decaf::net::Inet6Address::~~Inet6Address** () [virtual]

6.395.2 Friends And Related Function Documentation

- 6.395.2.1 **friend class InetAddress** [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**Inet6Address.h**

6.396 decaf::net::InetAddress Class Reference

Represents an IP address.

```
#include <src/main/decaf/net/InetAddress.h>
```

Inheritance diagram for decaf::net::InetAddress:

Public Member Functions

- virtual **~InetAddress** ()
- virtual **decaf::lang::ArrayPointer**< unsigned char > **getAddress** () const
Returns the Raw IP address in Network byte order.
- virtual std::string **getHostAddress** () const
Returns a textual representation of the IP Address.

- virtual std::string **getHostName** () const
*Get the host name associated with this **InetAddress** (p. 1884) instance.*
- virtual std::string **toString** () const
*Returns a string representation of the **InetAddress** (p. 1884) in the form 'hostname / ipaddress'.*
- virtual bool **isAnyLocalAddress** () const
*Check if this **InetAddress** (p. 1884) is a valid wildcard address.*
- virtual bool **isLoopbackAddress** () const
*Check if this **InetAddress** (p. 1884) is a valid loopback address.*
- virtual bool **isMulticastAddress** () const
*Check if this **InetAddress** (p. 1884) is a valid Multicast address.*
- virtual bool **isLinkLocalAddress** () const
*Check if this **InetAddress** (p. 1884) is a valid link local address.*
- virtual bool **isSiteLocalAddress** () const
*Check if this **InetAddress** (p. 1884) is a valid site local address.*
- virtual bool **isMCGlobal** () const
*Check if this **InetAddress** (p. 1884) is Multicast and has Global scope.*
- virtual bool **isMCNodeLocal** () const
*Check if this **InetAddress** (p. 1884) is Multicast and has Node Local scope.*
- virtual bool **isMCLinkLocal** () const
*Check if this **InetAddress** (p. 1884) is Multicast and has Link Local scope.*
- virtual bool **isMCSiteLocal** () const
*Check if this **InetAddress** (p. 1884) is Multicast and has Site Local scope.*
- virtual bool **isMCOrgLocal** () const
*Check if this **InetAddress** (p. 1884) is Multicast and has Organization scope.*

Static Public Member Functions

- static **InetAddress** **getByAddress** (const unsigned char *bytes, int numBytes)
*Given a raw IP Address in byte array form, create and return a new **InetAddress** (p. 1884) instance.*
- static **InetAddress** **getByAddress** (const std::string &hostname, const unsigned char *bytes, int numBytes)
*Given a host name and IPAddress return a new **InetAddress** (p. 1884).*
- static **InetAddress** **getLocalHost** ()
*Gets an **InetAddress** (p. 1884) that is the local host address.*

Protected Member Functions

- **InetAddress** ()
- **InetAddress** (const unsigned char *ipAddress, int numBytes)
- **InetAddress** (const std::string &hostname, const unsigned char *ipAddress, int numBytes)

Static Protected Member Functions

- static unsigned int **bytesToInt** (const unsigned char *bytes, int start)

Converts the bytes in an address array to an int starting from the value start treating the start value as the high order byte.

Protected Attributes

- std::string **hostname**
- bool **reached**
- decaf::lang::ArrayPointer< unsigned char > **addressBytes**

Static Protected Attributes

- static const unsigned char **loopbackBytes** [4]
- static const unsigned char **anyBytes** [4]
- static const **InetAddress ANY**
- static const **InetAddress LOOPBACK**

6.396.1 Detailed Description

Represents an IP address.

Since

1.0

6.396.2 Constructor & Destructor Documentation

6.396.2.1 `decaf::net::InetAddress::InetAddress ()` [protected]

6.396.2.2 `decaf::net::InetAddress::InetAddress (const unsigned char * ipAddress, int numBytes)` [protected]

6.396.2.3 `decaf::net::InetAddress::InetAddress (const std::string & hostname, const unsigned char * ipAddress, int numBytes)` [protected]

6.396.2.4 `virtual decaf::net::InetAddress::~~InetAddress ()` [virtual]

6.396.3 Member Function Documentation

6.396.3.1 `static unsigned int decaf::net::InetAddress::bytesToInt (const unsigned char * bytes, int start)` [static, protected]

Converts the bytes in an address array to an int starting from the value start treating the start value as the high order byte.

Parameters

bytes The array of bytes to convert to an int.

start The index in the array to treat as the high order byte.

Returns

an unsigned int that represents the address value.

6.396.3.2 `virtual decaf::lang::ArrayPointer<unsigned char> decaf::net::InetAddress::getAddress () const` [virtual]

Returns the Raw IP address in Network byte order.

The returned address is a copy of the bytes contained in this **InetAddress** (p.1884).

Returns

and ArrayPointer containing the raw bytes of the network address.

6.396.3.3 `static InetAddress decaf::net::InetAddress::getByAddress (const std::string & hostname, const unsigned char * bytes, int numBytes)` [static]

Given a host name and IPAddress return a new **InetAddress** (p.1884).

There is no name service checking or address validation done on the provided host name. The host name can either be machine name or the text based representation of the IP Address.

An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

Returns

a copy of an **InetAddress** (p.1884) that represents the given byte array address.

Exceptions

UnknownHostException (p. 3649) if the address array length is invalid.

6.396.3.4 `static InetAddress decaf::net::InetAddress::getByAddress (const unsigned char * bytes, int numBytes) [static]`

Given a raw IP Address in byte array form, create and return a new **InetAddress** (p.1884) instance.

An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

Returns

a copy of an **InetAddress** (p.1884) that represents the given byte array address.

Exceptions

UnknownHostException (p. 3649) if the address array length is invalid.

6.396.3.5 `virtual std::string decaf::net::InetAddress::getHostAddress () const [virtual]`

Returns a textual representation of the IP Address.

Returns

the string form of the IP Address.

6.396.3.6 `virtual std::string decaf::net::InetAddress::getHostName () const [virtual]`

Get the host name associated with this **InetAddress** (p.1884) instance.

If a host name was set upon construction then that value is returned, otherwise a reverse name lookup will be performed to attempt to get the host name associated with the set IP Address. If the host name cannot be resolved the textual representation of the IP Address is returned instead.

Returns

the name of the host associated with this set IP Address.

6.396.3.7 `static InetAddress decaf::net::InetAddress::getLocalHost () [static]`

Gets an **InetAddress** (p.1884) that is the local host address.

If the localhost value cannot be resolved then the **InetAddress** (p.1884) for Loopback is returned.

Returns

a new **InetAddress** (p.1884) object that contains the local host address.

Exceptions

UnknownHostException (p. 3649) if the address for local host is not found.

6.396.3.8 `virtual bool decaf::net::InetAddress::isAnyLocalAddress () const`
[inline, virtual]

Check if this **InetAddress** (p. 1884) is a valid wildcard address.

Returns

true if the address is a wildcard address.

Reimplemented in **decaf::net::Inet4Address** (p. 1881).

6.396.3.9 `virtual bool decaf::net::InetAddress::isLinkLocalAddress () const`
[inline, virtual]

Check if this **InetAddress** (p. 1884) is a valid link local address.

Returns

true if the address is a link local address.

Reimplemented in **decaf::net::Inet4Address** (p. 1881).

6.396.3.10 `virtual bool decaf::net::InetAddress::isLoopbackAddress () const`
[inline, virtual]

Check if this **InetAddress** (p. 1884) is a valid loopback address.

Returns

true if the address is a loopback address.

Reimplemented in **decaf::net::Inet4Address** (p. 1882).

6.396.3.11 `virtual bool decaf::net::InetAddress::isMCGlobal () const` [inline, virtual]

Check if this **InetAddress** (p. 1884) is Multicast and has Global scope.

Returns

true if the address is Multicast and has Global scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1882).

6.396.3.12 `virtual bool decaf::net::InetAddress::isMCLinkLocal () const`
[inline, virtual]

Check if this **InetAddress** (p.1884) is Multicast and has Link Local scope.

Returns

true if the address is Multicast and has Link Local scope.

Reimplemented in **decaf::net::Inet4Address** (p.1882).

6.396.3.13 `virtual bool decaf::net::InetAddress::isMCNodeLocal () const`
[inline, virtual]

Check if this **InetAddress** (p.1884) is Multicast and has Node Local scope.

Returns

true if the address is Multicast and has Node Local scope.

Reimplemented in **decaf::net::Inet4Address** (p.1882).

6.396.3.14 `virtual bool decaf::net::InetAddress::isMCOrgLocal () const` [inline, virtual]

Check if this **InetAddress** (p.1884) is Multicast and has Organization scope.

Returns

true if the address is Multicast and has Organization scope.

Reimplemented in **decaf::net::Inet4Address** (p.1882).

6.396.3.15 `virtual bool decaf::net::InetAddress::isMCSiteLocal () const`
[inline, virtual]

Check if this **InetAddress** (p.1884) is Multicast and has Site Local scope.

Returns

true if the address is Multicast and has Site Local scope.

Reimplemented in **decaf::net::Inet4Address** (p.1883).

6.396.3.16 `virtual bool decaf::net::InetAddress::isMulticastAddress () const`
[inline, virtual]

Check if this **InetAddress** (p.1884) is a valid Multicast address.

Returns

true if the address is a Multicast address.

Reimplemented in **decaf::net::Inet4Address** (p.1883).

6.396.3.17 `virtual bool decaf::net::InetAddress::isSiteLocalAddress () const`
`[inline, virtual]`

Check if this **InetAddress** (p. 1884) is a valid site local address.

Returns

true if the address is a site local address.

Reimplemented in **decaf::net::Inet4Address** (p. 1883).

6.396.3.18 `virtual std::string decaf::net::InetAddress::toString () const`
`[virtual]`

Returns a string representation of the **InetAddress** (p. 1884) in the form 'hostname / ipaddress'.

If the hostname is not resolved than it appears as empty.

Returns

string value of this **InetAddress** (p. 1884).

6.396.4 Field Documentation

6.396.4.1 `decaf::lang::ArrayPointer<unsigned char> decaf::net::InetAddress::addressBytes` `[mutable, protected]`

6.396.4.2 `const InetAddress decaf::net::InetAddress::ANY` `[static, protected]`

6.396.4.3 `const unsigned char decaf::net::InetAddress::anyBytes[4]` `[static, protected]`

6.396.4.4 `std::string decaf::net::InetAddress::hostname` `[mutable, protected]`

6.396.4.5 `const InetAddress decaf::net::InetAddress::LOOPBACK` `[static, protected]`

6.396.4.6 `const unsigned char decaf::net::InetAddress::loopbackBytes[4]` `[static, protected]`

6.396.4.7 `bool decaf::net::InetAddress::reached` `[mutable, protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/net/InetAddress.h`

6.397 decaf::net::InetSocketAddress Class Reference

```
#include <src/main/decaf/net/InetSocketAddress.h>
```

Inheritance diagram for **decaf::net::InetSocketAddress**:

Public Member Functions

- **InetSocketAddress** ()
- virtual **~InetSocketAddress** ()

6.397.1 Constructor & Destructor Documentation

6.397.1.1 **decaf::net::InetSocketAddress::InetSocketAddress** ()

6.397.1.2 **virtual decaf::net::InetSocketAddress::~~InetSocketAddress** ()
[virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/InetSocketAddress.h`

6.398 inflate__state Struct Reference

```
#include <src/main/decaf/internal/util/zip/inflate.h>
```

Data Fields

- **inflate__mode** mode
- int **last**
- int **wrap**
- int **havedict**
- int **flags**
- unsigned **dmax**
- unsigned long **check**
- unsigned long **total**
- **gz_headerp** head
- unsigned **wbits**
- unsigned **wsize**
- unsigned **whave**
- unsigned **wnext**
- unsigned char FAR * **window**
- unsigned long **hold**
- unsigned **bits**
- unsigned **length**
- unsigned **offset**
- unsigned **extra**
- **code** const FAR * **lencode**
- **code** const FAR * **distcode**
- unsigned **lenbits**
- unsigned **distbits**
- unsigned **ncode**
- unsigned **nlen**
- unsigned **ndist**
- unsigned **have**

- code FAR * **next**
- unsigned short **lens** [320]
- unsigned short **work** [288]
- code **codes** [ENOUGH]
- int **sane**
- int **back**
- unsigned **was**

6.398.1 Field Documentation

- 6.398.1.1 `int inflate__state::back`
- 6.398.1.2 `unsigned inflate__state::bits`
- 6.398.1.3 `unsigned long inflate__state::check`
- 6.398.1.4 `code inflate__state::codes[ENOUGH]`
- 6.398.1.5 `unsigned inflate__state::distbits`
- 6.398.1.6 `code const FAR* inflate__state::distcode`
- 6.398.1.7 `unsigned inflate__state::dmax`
- 6.398.1.8 `unsigned inflate__state::extra`
- 6.398.1.9 `int inflate__state::flags`
- 6.398.1.10 `unsigned inflate__state::have`
- 6.398.1.11 `int inflate__state::havedict`
- 6.398.1.12 `gz_headerp inflate__state::head`
- 6.398.1.13 `unsigned long inflate__state::hold`
- 6.398.1.14 `int inflate__state::last`
- 6.398.1.15 `unsigned inflate__state::lenbits`
- 6.398.1.16 `code const FAR* inflate__state::lencode`
- 6.398.1.17 `unsigned inflate__state::length`
- 6.398.1.18 `unsigned short inflate__state::lens[320]`
- 6.398.1.19 `inflate__mode inflate__state::mode`
- 6.398.1.20 `unsigned inflate__state::ncode`
- 6.398.1.21 `unsigned inflate__state::ndist`
- 6.398.1.22 `code FAR* inflate__state::next`
- 6.398.1.23 `unsigned inflate__state::nlen`
- 6.398.1.24 `unsigned inflate__state::offset`
- 6.398.1.25 `int inflate__state::sane`
- 6.398.1.26 `unsigned long inflate__state::total`
- 6.398.1.27 `unsigned inflate__state::was`
- 6.398.1.28 `unsigned inflate__state::wbits`
- 6.398.1.29 `unsigned inflate__state::whave`
- 6.398.1.30 `unsigned char FAR* inflate__state::window`

- `src/main/decaf/internal/util/zip/inflate.h`

6.399 decaf::util::zip::Inflater Class Reference

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see specification).

```
#include <src/main/decaf/util/zip/Inflater.h>
```

Public Member Functions

- **Inflater** ()
Creates a new decompressor.
- **Inflater** (bool nowrap)
Creates a new decompressor.
- virtual **~Inflater** ()
- void **setInput** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)
Sets input data for decompression.
- void **setInput** (const std::vector< unsigned char > &buffer, int offset, int length) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException)
Sets input data for decompression.
- void **setInput** (const std::vector< unsigned char > &buffer) throw (decaf::lang::exceptions::IllegalStateException)
Sets input data for decompression.
- int **getRemaining** () const
Returns the total number of bytes remaining in the input buffer.
- void **setDictionary** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Sets the preset dictionary to the given array of bytes.
- void **setDictionary** (const std::vector< unsigned char > &buffer, int offset, int length) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Sets the preset dictionary to the given array of bytes.
- void **setDictionary** (const std::vector< unsigned char > &buffer) throw (decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IllegalArgumentException)

Sets the preset dictionary to the given array of bytes.

- bool **needsInput** () const
- bool **needsDictionary** () const
- void **finish** ()

When called, indicates that decompression should end with the current contents of the input buffer.

- bool **finished** () const
- int **inflate** (unsigned char *buffer, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException)

Uncompresses bytes into specified buffer.

- int **inflate** (std::vector< unsigned char > &buffer, int offset, int length) throw (decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException)

Uncompresses bytes into specified buffer.

- int **inflate** (std::vector< unsigned char > &buffer) throw (decaf::lang::exceptions::IllegalStateException, decaf::util::zip::DataFormatException)

Uncompresses bytes into specified buffer.

- long long **getAdler** () const throw (decaf::lang::exceptions::IllegalStateException)
- long long **getBytesRead** () const throw (decaf::lang::exceptions::IllegalStateException)
- long long **getBytesWritten** () const throw (decaf::lang::exceptions::IllegalStateException)
- void **reset** () throw (decaf::lang::exceptions::IllegalStateException)

Resets deflater so that a new set of input data can be processed.

- void **end** ()

Closes the decompressor and discards any unprocessed input.

6.399.1 Detailed Description

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see [specification](#)). Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **InflaterInputStream** (p. 1902) and its descendants.

The typical usage of a **Inflater** (p. 1894) outside this package consists of a specific call to one of its constructors before being passed to an instance of **InflaterInputStream** (p. 1902).

See also

InflaterInputStream (p. 1902)
Deflater (p. 1595)

Since

1.0

6.399.2 Constructor & Destructor Documentation

6.399.2.1 decaf::util::zip::Inflater::Inflater ()

Creates a new decompressor.

This constructor defaults the inflater to use the ZLIB header and checksum fields.

6.399.2.2 decaf::util::zip::Inflater::Inflater (bool nowrap)

Creates a new decompressor.

If the parameter 'nowrap' is true then the ZLIB header and checksum fields will not be used. This provides compatibility with the compression format used by both GZIP and PKZIP.

Note: When using the 'nowrap' option it is also necessary to provide an extra "dummy" byte as input. This is required by the ZLIB native library in order to support certain optimizations.

6.399.2.3 virtual decaf::util::zip::Inflater::~Inflater () [virtual]

6.399.3 Member Function Documentation

6.399.3.1 void decaf::util::zip::Inflater::end ()

Closes the decompressor and discards any unprocessed input.

This method should be called when the decompressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Inflater** (p.1894) object is undefined.

6.399.3.2 void decaf::util::zip::Inflater::finish ()

When called, indicates that decompression should end with the current contents of the input buffer.

6.399.3.3 bool decaf::util::zip::Inflater::finished () const

Returns

true if the end of the compressed data output stream has been reached.

6.399.3.4 long long decaf::util::zip::Inflater::getAdler () const throw (decaf::lang::exceptions::IllegalStateException)

Returns

the ADLER-32 value of the uncompressed data.

Exceptions

IllegalStateException if in the end state.

6.399.3.5 `long long decaf::util::zip::Inflater::getBytesRead () const throw (decaf::lang::exceptions::IllegalStateException)`

Returns

the total number of compressed bytes input so far.

Exceptions

IllegalStateException if in the end state.

6.399.3.6 `long long decaf::util::zip::Inflater::getBytesWritten () const throw (decaf::lang::exceptions::IllegalStateException)`

Returns

the total number of decompressed bytes output so far.

Exceptions

IllegalStateException if in the end state.

6.399.3.7 `int decaf::util::zip::Inflater::getRemaining () const`

Returns the total number of bytes remaining in the input buffer.

This can be used to find out what bytes still remain in the input buffer after decompression has finished.

Returns

the total number of bytes remaining in the input buffer

6.399.3.8 `int decaf::util::zip::Inflater::inflate (std::vector< unsigned char > & buffer) throw (decaf::lang::exceptions::IllegalStateException, decaf::util::zip::DataFormatException)`

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p.1899) or **needsDictionary()** (p.1899) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p.1896) can be used to get the Adler-32 value of the dictionary required.

Parameters

buffer The Buffer to write the compressed data to.

Exceptions

IllegalStateException if in the end state.

DataFormatException (p. 1450) if the compressed data format is invalid.

```

6.399.3.9  int decaf::util::zip::Inflater::inflate ( std::vector<
            unsigned char > & buffer, int offset, int length )
            throw ( decaf::lang::exceptions::IllegalStateException,
                    decaf::lang::exceptions::IndexOutOfBoundsException,
                    decaf::util::zip::DataFormatException )

```

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p.1899) or **needsDictionary()** (p.1899) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p.1896) can be used to get the Adler-32 value of the dictionary required.

Parameters

buffer The Buffer to write the compressed data to.

offset The position in the Buffer to start writing at.

length The maximum number of byte of data to write.

Exceptions

IllegalStateException if in the end state.

IndexOutOfBoundsException if the offset + length > size of the buffer.

DataFormatException (p. 1450) if the compressed data format is invalid.

```

6.399.3.10 int decaf::util::zip::Inflater::inflate ( unsigned char
            * buffer, int size, int offset, int length )
            throw ( decaf::lang::exceptions::NullPointerException,
                    decaf::lang::exceptions::IllegalStateException,
                    decaf::lang::exceptions::IndexOutOfBoundsException,
                    decaf::util::zip::DataFormatException )

```

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p.1899) or **needsDictionary()** (p.1899) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p.1896) can be used to get the Adler-32 value of the dictionary required.

Parameters

buffer The Buffer to write the compressed data to.

size The size of the buffer passed in.

offset The position in the Buffer to start writing at.

length The maximum number of byte of data to write.

Exceptions

NullPointerException if buffer is NULL.

IllegalStateException if in the end state.

IndexOutOfBoundsException if the offset + length > size of the buffer.

DataFormatException (p. 1450) if the compressed data format is invalid.

6.399.3.11 `bool decaf::util::zip::Inflater::needsDictionary () const`**Returns**

true if a preset dictionary is needed for decompression.

6.399.3.12 `bool decaf::util::zip::Inflater::needsInput () const`**Returns**

true if the input data buffer is empty and **setInput()** (p. 1901) should be called in order to provide more input

6.399.3.13 `void decaf::util::zip::Inflater::reset () throw (decaf::lang::exceptions::IllegalStateException)`

Resets deflater so that a new set of input data can be processed.

Keeps current decompression level and strategy settings.

Exceptions

IllegalStateException if in the end state.

6.399.3.14 `void decaf::util::zip::Inflater::setDictionary (const std::vector< unsigned char > & buffer, int offset, int length) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 1898) returns 0 and **needsDictionary()** (p. 1899) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 1896) can be used to get the Adler-32 value of the dictionary needed.

Parameters

buffer The Buffer to read in for decompression.

offset The position in the Buffer to start reading from.

length The number of bytes to read from the input buffer.

Exceptions

IndexOutOfBoundsException if the offset + length > size of the buffer.

IllegalStateException if in the end state.

IllegalArgumentException if the given dictionary doesn't match the required dictionaries checksum value.

6.399.3.15 void decaf::util::zip::Inflater::setDictionary (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p.1898) returns 0 and **needsDictionary()** (p.1899) returns true indicating that a preset dictionary is required. The method **getAdler()** (p.1896) can be used to get the Adler-32 value of the dictionary needed.

Parameters

buffer The Buffer to read in for decompression.

size The size of the buffer passed in.

offset The position in the Buffer to start reading from.

length The number of bytes to read from the input buffer.

Exceptions

NullPointerException if buffer is NULL.

IndexOutOfBoundsException if the offset + length > size of the buffer.

IllegalStateException if in the end state.

IllegalArgumentException if the given dictionary doesn't match the required dictionaries checksum value.

6.399.3.16 void decaf::util::zip::Inflater::setDictionary (const std::vector< unsigned char > & *buffer*) throw (decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IllegalArgumentException)

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p.1898) returns 0 and **needsDictionary()** (p.1899) returns true indicating that a preset dictionary is required. The method **getAdler()** (p.1896) can be used to get the Adler-32 value of the dictionary needed.

Parameters

buffer The Buffer to read in for decompression.

Exceptions

IllegalStateException if in the end state.

IllegalArgumentException if the given dictionary doesn't match the required dictionaries checksum value.

```

6.399.3.17 void decaf::util::zip::Inflater::setInput ( const unsigned
char * buffer, int size, int offset, int length )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::IllegalStateException )

```

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1899) returns true indicating that more input data is required.

Parameters

buffer The Buffer to read in for decompression.

size The size of the buffer passed in.

offset The position in the Buffer to start reading from.

length The number of bytes to read from the input buffer.

Exceptions

NullPointerException if buffer is NULL.

IndexOutOfBoundsException if the offset + length > size of the buffer.

IllegalStateException if in the end state.

```

6.399.3.18 void decaf::util::zip::Inflater::setInput ( const std::vector<
unsigned char > & buffer, int offset, int length )
throw ( decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::IllegalStateException )

```

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1899) returns true indicating that more input data is required.

Parameters

buffer The Buffer to read in for decompression.

offset The position in the Buffer to start reading from.

length The number of bytes to read from the input buffer.

Exceptions

IndexOutOfBoundsException if the offset + length > size of the buffer.

IllegalStateException if in the end state.

```

6.399.3.19 void decaf::util::zip::Inflater::setInput ( const std::vector< unsigned
char > & buffer ) throw ( decaf::lang::exceptions::IllegalStateException
)

```

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1899) returns true indicating that more input data is required.

Parameters

buffer The Buffer to read in for decompression.

Exceptions

IllegalStateException if in the end state.

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Inflater.h**

6.400 decaf::util::zip::InflaterInputStream Class Reference

A FilterInputStream that decompresses data read from the wrapped InputStream instance.

```
#include <src/main/decaf/util/zip/InflaterInputStream.h>
```

Inheritance diagram for decaf::util::zip::InflaterInputStream:

Public Member Functions

- **InflaterInputStream** (decaf::io::InputStream *inputStream, bool own=false)
Create an instance of this class with a default inflater and buffer size.
- **InflaterInputStream** (decaf::io::InputStream *inputStream, Inflater *inflater, bool own=false)
*Creates a new **InflaterInputStream** (p. 1902) with a user supplied **Inflater** (p. 1894) and a default buffer size.*
- **InflaterInputStream** (decaf::io::InputStream *inputStream, Inflater *inflater, int bufferSize, bool own=false)
*Creates a new DeflateOutputStream with a user supplied **Inflater** (p. 1894) and specified buffer size.*
- virtual ~**InflaterInputStream** ()
- virtual int **available** () const throw (decaf::io::IOException)
Indicates the number of bytes available.
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.
The default implementation of this method returns zero.
Returns
the number of bytes available on this input stream.
Exceptions
IOException (p. 2003) if an I/O error occurs.
- virtual void **close** () throw (decaf::io::IOException)

*Closes the **InputStream** (p. 1909) freeing any resources that might have been acquired during the lifetime of this stream.*

The default implementation of this method does nothing.

- virtual long long **skip** (long long num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Skips over and discards n bytes of data from this input stream.

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

***IOException** (p. 2003) if an I/O error occurs.*

***UnsupportedOperationException** if the concrete stream class does not support skipping bytes.*

- virtual void **mark** (int readLimit)

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit The max bytes read before marked position is invalid.

- virtual void **reset** () throw (decaf::io::IOException)

Repositions this stream to the position at the time the mark method was last called on this input stream.

*If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2003) might be thrown. * If such an **IOException** (p. 2003) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 2003). * If an **IOException** (p. 2003) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 2003).*

Exceptions

***IOException** (p. 2003) if an I/O error occurs.*

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Protected Member Functions

- virtual void **fill** () throw (decaf::io::IOException)
Fills the input buffer with the next chunk of data.
- virtual int **doReadByte** () throw (decaf::io::IOException)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int **length**) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

Protected Attributes

- **Inflater * inflater**
*The **Inflater** (p. 1894) instance to use.*
- std::vector< unsigned char > **buff**
The buffer to hold chunks of data read from the stream before inflation.
- int **length**
The amount of data currently stored in the input buffer.
- bool **ownInflater**
- bool **atEOF**

Static Protected Attributes

- static const int **DEFAULT_BUFFER_SIZE**

6.400.1 Detailed Description

A FilterInputStream that decompresses data read from the wrapped InputStream instance.

Since

1.0

6.400.2 Constructor & Destructor Documentation

- 6.400.2.1** decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream * *inputStream*, bool *own* = *false*)

Create an instance of this class with a default inflater and buffer size.

Parameters

inputStream The **InputStream** instance to wrap.

own Should this **Filter** take ownership of the **InputStream** pointer (defaults to false).

6.400.2.2 `decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream * inputStream, Inflater * inflater, bool own = false)`

Creates a new **InflaterInputStream** (p.1902) with a user supplied **Inflater** (p.1894) and a default buffer size.

When the user supplied a **Inflater** (p.1894) instance the **InflaterInputStream** (p.1902) does not take ownership of the **Inflater** (p.1894) pointer, the caller is still responsible for deleting the **Inflater** (p.1894).

Parameters

inputStream The **InputStream** instance to wrap.

inflater The user supplied **Inflater** (p.1894) to use for decompression. (

own Should this filter take ownership of the **InputStream** pointer (default is false).

Exceptions

NullPointerException if the **Inflater** (p.1894) given is NULL.

6.400.2.3 `decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream * inputStream, Inflater * inflater, int bufferSize, bool own = false)`

Creates a new **DeflateOutputStream** with a user supplied **Inflater** (p.1894) and specified buffer size.

When the user supplied a **Inflater** (p.1894) instance the **InflaterInputStream** (p.1902) does not take ownership of the **Inflater** (p.1894) pointer, the caller is still responsible for deleting the **Inflater** (p.1894).

Parameters

inputStream The **InputStream** instance to wrap.

inflater The user supplied **Inflater** (p.1894) to use for decompression.

bufferSize The size of the input buffer.

own Should this filter take ownership of the **InputStream** pointer (default is false).

Exceptions

NullPointerException if the **Inflater** (p.1894) given is NULL.

IllegalArgumentException if the *bufferSize* value is zero.

6.400.2.4 `virtual decaf::util::zip::InflaterInputStream::~~InflaterInputStream ()`
[virtual]

6.400.3 Member Function Documentation

6.400.3.1 `virtual int decaf::util::zip::InflaterInputStream::available () const`
`throw (decaf::io::IOException)` [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

Until EOF this method always returns 1, thereafter it always returns 0.

Reimplemented from `decaf::io::FilterInputStream` (p.1773).

6.400.3.2 `virtual void decaf::util::zip::InflaterInputStream::close () throw (`
`decaf::io::IOException)` [virtual]

Closes the **InputStream** (p. 1909) freeing any resources that might have been aquired during the lifetime of this stream.

The default implementation of this method does nothing.

Closes any resources associated with this **InflaterInputStream** (p. 1902).

Reimplemented from `decaf::io::FilterInputStream` (p.1774).

6.400.3.3 `virtual int decaf::util::zip::InflaterInputStream::doReadArrayBounded`
`(unsigned char * buffer, int size, int offset,`
`int length) throw (decaf::io::IOException,`
`decaf::lang::exceptions::IndexOutOfBoundsException,`
`decaf::lang::exceptions::NullPointerException)` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p.1774).

6.400.3.4 `virtual int decaf::util::zip::InflaterInputStream::doReadByte () throw`
`(decaf::io::IOException)` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p.1774).

6.400.3.5 `virtual void decaf::util::zip::InflaterInputStream::fill () throw (decaf::io::IOException)` [protected, virtual]

Fills the input buffer with the next chunk of data.

Exceptions

IOException if an I/O error occurs.

6.400.3.6 `virtual void decaf::util::zip::InflaterInputStream::mark (int readLimit)` [virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit The max bytes read before marked position is invalid.

Does nothing.

Reimplemented from `decaf::io::FilterInputStream` (p.1775).

6.400.3.7 `virtual bool decaf::util::zip::InflaterInputStream::markSupported () const` [virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Always returns false.

Reimplemented from `decaf::io::FilterInputStream` (p.1775).

6.400.3.8 `virtual void decaf::util::zip::InflaterInputStream::reset () throw (decaf::io::IOException)` [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called

is larger than the argument to mark at that last call, then an **IOException** (p. 2003) might be thrown. * If such an **IOException** (p. 2003) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 2003). * If an **IOException** (p. 2003) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2003).

Exceptions

IOException (p. 2003) if an I/O error occurs.

Always throws an **IOException** when called.

Reimplemented from **decaf::io::FilterInputStream** (p. 1775).

6.400.3.9 virtual long long decaf::util::zip::InflaterInputStream::skip
 (long long *num*) throw (decaf::io::IOException,
 decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

Skips the specified amount of uncompressed input data.

Reimplemented from **decaf::io::FilterInputStream** (p. 1776).

6.400.4 Field Documentation

6.400.4.1 `bool decaf::util::zip::InflaterInputStream::atEOF` [protected]

6.400.4.2 `std::vector<unsigned char> decaf::util::zip::InflaterInputStream::buff`
[protected]

The buffer to hold chunks of data read from the stream before inflation.

6.400.4.3 `const int decaf::util::zip::InflaterInputStream::DEFAULT_BUFFER_SIZE` [static, protected]

6.400.4.4 `Inflater* decaf::util::zip::InflaterInputStream::inflater` [protected]

The **Inflater** (p.1894) instance to use.

6.400.4.5 `int decaf::util::zip::InflaterInputStream::length` [protected]

The amount of data currently stored in the input buffer.

6.400.4.6 `bool decaf::util::zip::InflaterInputStream::ownInflater` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/InflaterInputStream.h`

6.401 decaf::io::InputStream Class Reference

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

```
#include <src/main/decaf/io/InputStream.h>
```

Inheritance diagram for decaf::io::InputStream:

Public Member Functions

- **InputStream** ()
- virtual **~InputStream** ()
- virtual void **close** () throw (decaf::io::IOException)
*Closes the **InputStream** (p.1909) freeing any resources that might have been aquired during the lifetime of this stream.*
- virtual void **mark** (int readLimit)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (decaf::io::IOException)

Repositions this stream to the position at the time the mark method was last called on this input stream.

- virtual bool **markSupported** () const
Determines if this input stream supports the mark and reset methods.
- virtual int **available** () const throw (decaf::io::IOException)
Indicates the number of bytes available.
- virtual int **read** () throw (decaf::io::IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
Reads up to size bytes of data from the input stream into an array of bytes.
- virtual int **read** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)
Reads up to length bytes of data from the input stream into an array of bytes.
- virtual long long **skip** (long long num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.
- virtual std::string **toString** () const
Output a String representation of this object.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to `Notify`.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

Protected Member Functions

- virtual int **doReadByte** ()=0 throw (decaf::io::IOException)
- virtual int **doReadArray** (unsigned char *buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

6.401.1 Detailed Description

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

Since

1.0

6.401.2 Constructor & Destructor Documentation

6.401.2.1 decaf::io::InputStream::InputStream ()

6.401.2.2 virtual decaf::io::InputStream::~~InputStream () [virtual]

6.401.3 Member Function Documentation

6.401.3.1 virtual int decaf::io::InputStream::available () const throw (decaf::io::IOException) [inline, virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented in `decaf::internal::io::StandardInputStream` (p. 3353), `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream` (p. 2697), `decaf::internal::net::tcp::TcpSocketInputStream` (p. 3507), `decaf::io::BlockingByteArrayInputStream` (p. 773), `decaf::io::BufferedInputStream` (p. 864), `decaf::io::ByteArrayInputStream` (p. 947), `decaf::io::FilterInputStream` (p. 1773), `decaf::io::PushbackInputStream` (p. 2943), and `decaf::util::zip::InflaterInputStream` (p. 1906).

6.401.3.2 virtual void decaf::io::InputStream::close () throw (decaf::io::IOException) [virtual]

Closes the **InputStream** (p. 1909) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented in `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream` (p. 2697), `decaf::internal::net::tcp::TcpSocketInputStream` (p. 3508), `decaf::io::BlockingByteArrayInputStream` (p. 773), `decaf::io::BufferedInputStream` (p. 864), `decaf::io::FilterInputStream` (p. 1774), and `decaf::util::zip::InflaterInputStream` (p. 1906).

6.401.3.3 virtual int decaf::io::InputStream::doReadArray (unsigned char * *buffer*, int *size*) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException) [protected, virtual]

Reimplemented in `decaf::io::FilterInputStream` (p. 1774).

6.401.3.4 virtual int decaf::io::InputStream::doReadArrayBounded (unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException) [protected, virtual]

Reimplemented in `activemq::io::LoggingInputStream` (p. 2250), `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream` (p. 2697), `decaf::internal::net::tcp::TcpSocketInputStream` (p. 3508), `decaf::io::BlockingByteArrayInputStream` (p. 773), `decaf::io::BufferedInputStream` (p. 865), `decaf::io::ByteArrayInputStream` (p. 948), `decaf::io::FilterInputStream` (p. 1774), `decaf::io::PushbackInputStream` (p. 2943), and `decaf::util::zip::CheckedInputStream` (p. 1057), and `decaf::util::zip::InflaterInputStream` (p. 1906).

6.401.3.5 virtual int decaf::io::InputStream::doReadByte () throw (decaf::io::IOException) [protected, pure virtual]

Implemented in `activemq::io::LoggingInputStream` (p. 2251), `decaf::internal::io::StandardInputStream` (p. 3353), `decaf::internal::net::ssl::openssl::OpenSSLSocketInput`

(p. 2698), `decaf::internal::net::tcp::TcpSocketInputStream` (p. 3508), `decaf::io::BlockingByteArrayInputStream` (p. 773), `decaf::io::BufferedInputStream` (p. 865), `decaf::io::ByteArrayInputStream` (p. 948), `decaf::io::FilterInputStream` (p. 1774), `decaf::io::PushbackInputStream` (p. 2943), `decaf::util::zip::CheckedInputStream` (p. 1057), and `decaf::util::zip::InflaterInputStream` (p. 1906).

6.401.3.6 `virtual void decaf::io::InputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3463).

6.401.3.7 `virtual void decaf::io::InputStream::mark (int readLimit) [virtual]`

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit The max bytes read before marked position is invalid.

Reimplemented in `decaf::io::BufferedInputStream` (p. 865), `decaf::io::ByteArrayInputStream` (p. 948), `decaf::io::FilterInputStream` (p. 1775), `decaf::io::PushbackInputStream` (p. 2944), and `decaf::util::zip::InflaterInputStream` (p. 1907).

6.401.3.8 `virtual bool decaf::io::InputStream::markSupported () const [inline, virtual]`

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented in `decaf::io::BufferedInputStream` (p. 865), `decaf::io::ByteArrayInputStream` (p. 948), `decaf::io::FilterInputStream` (p. 1775), `decaf::io::PushbackInputStream` (p. 2944), and `decaf::util::zip::InflaterInputStream` (p. 1907).

6.401.3.9 `virtual void decaf::io::InputStream::notify ()
 throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [inline,
 virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3464).

6.401.3.10 `virtual void decaf::io::InputStream::notifyAll ()
 throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [inline,
 virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3465).

6.401.3.11 `virtual int decaf::io::InputStream::read (unsigned char * buffer,
 int size, int offset, int length) throw (decaf::io::IOException,
 decaf::lang::exceptions::IndexOutOfBoundsException,
 decaf::lang::exceptions::NullPointerException) [virtual]`

Reads up to length bytes of data from the input stream into an array of bytes.

An attempt is made to read as many as length bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If length is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into b.

The first byte read is stored into element b[off], the next one into b[off+1], and so on. The number of bytes read is, at most, equal to length. Let k be the number of bytes actually read; these bytes will be stored in elements b[off] through b[off+k-1], leaving elements b[offset+k] through b[offset+length-1] unaffected.

In every case, elements `b[0]` through `b[offset]` and elements `b[offset+length]` through `b[b.length-1]` are unaffected.

This method called the protected virtual method `doReadArrayBounded` which simply calls the method `doReadByte()` (p.1912) repeatedly. If the first such call results in an **IOException** (p.2003), that exception is returned. If any subsequent call to `doReadByte()` (p.1912) results in a **IOException** (p.2003), the exception is caught and treated as if it were end of file; the bytes read up to that point are stored into the buffer and the number of bytes read before the exception occurred is returned. The default implementation of this method blocks until the requested amount of input data has been read, end of file is detected, or an exception is thrown. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

- buffer* The target buffer to write the read in data to.
- size* The size in bytes of the target buffer.
- offset* The position in the buffer to start inserting the read in data.
- length* The maximum number of bytes that should be read into buffer.

Returns

The number of bytes read or -1 if EOF is detected

Exceptions

- IOException** (p. 2003) if an I/O error occurs.
- NullPointerException** if buffer passed is NULL.
- IndexOutOfBoundsException** if `length > size - offset`.

6.401.3.12 `virtual int decaf::io::InputStream::read () throw (decaf::io::IOException)` [virtual]

Reads a single byte from the buffer.

The value byte is returned as an int in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown.

The default implementation of this method calls the internal virtual method `doReadByte` which is a pure virtual method and must be overridden by all subclasses.

Returns

The next byte or -1 if the end of stream is reached.

Exceptions

- IOException** (p. 2003) if an I/O error occurs.

6.401.3.13 `virtual int decaf::io::InputStream::read (unsigned char * buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)` [virtual]

Reads up to size bytes of data from the input stream into an array of bytes.

An attempt is made to read as many as size bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If size is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into b.

This method called the protected virtual method doReadArray which by default is the same as calling read(buffer, size, 0, size). Subclasses can customize the behavior of this method by overriding the doReadArray method to provide a better performing read operation.

Parameters

buffer The target buffer to write the read in data to.

size The size in bytes of the target buffer.

Returns

The number of bytes read or -1 if EOF is detected

Exceptions

IOException (p. 2003) if an I/O error occurs.

NullPointerException if buffer passed is NULL.

6.401.3.14 virtual void decaf::io::InputStream::reset () throw (decaf::io::IOException) [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2003) might be thrown. * If such an **IOException** (p. 2003) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 2003). * If an **IOException** (p. 2003) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2003).

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented in **decaf::io::BufferedInputStream** (p. 865), **decaf::io::ByteArrayInputStream** (p. 949), **decaf::io::FilterInputStream** (p. 1775), **decaf::io::PushbackInputStream** (p. 2944), and **decaf::util::zip::InflaterInputStream** (p. 1907).

6.401.3.15 `virtual long long decaf::io::InputStream::skip (long long num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)` [virtual]

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2698), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3508), **decaf::io::BlockingByteArrayInputStream** (p. 774), **decaf::io::BufferedInputStream** (p. 866), **decaf::io::ByteArrayInputStream** (p. 950), **decaf::io::FilterInputStream** (p. 1776), **decaf::io::PushbackInputStream** (p. 2945), **decaf::util::zip::CheckedInputStream** (p. 1057), and **decaf::util::zip::InflaterInputStream** (p. 1908).

6.401.3.16 `virtual std::string decaf::io::InputStream::toString () const` [virtual]

Output a String representation of this object.

The default version of this method just prints the Class Name.

Returns

a string representation of the object.

6.401.3.17 `virtual bool decaf::io::InputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException)` [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3466).

6.401.3.18 virtual void decaf::io::InputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3467).

6.401.3.19 virtual void decaf::io::InputStream::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3470).

6.401.3.20 virtual void decaf::io::InputStream::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3468).

```
6.401.3.21 virtual void decaf::io::InputStream::wait ( long long millisecs,
int nanos ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3471).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/InputStream.h`

6.402 decaf::io::InputStreamReader Class Reference

An **InputStreamReader** (p. 1919) is a bridge from byte streams to character streams.

```
#include <src/main/decaf/io/InputStreamReader.h>
```

Inheritance diagram for decaf::io::InputStreamReader:

Public Member Functions

- **InputStreamReader** (**InputStream** *stream, bool own=false)
Create a new *InputStreamReader* (p. 1919) that wraps the given *InputStream* (p. 1909).
- virtual ~**InputStreamReader** ()
- virtual void **close** () throw (decaf::io::IOException)

- virtual bool **ready** () const throw (decaf::io::IOException)

Tells whether this stream is ready to be read.

Protected Member Functions

- virtual int **doReadArrayBounded** (char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

Override this method to customize the functionality of the method read(unsigned char buffer, int size, int offset, int length).*

- virtual void **checkClosed** () const throw (decaf::io::IOException)

6.402.1 Detailed Description

An **InputStreamReader** (p. 1919) is a bridge from byte streams to character streams. For top efficiency, consider wrapping an **InputStreamReader** (p. 1919) within a **BufferedReader**. For example:

```
BufferedReader* in = new BufferedReader( new InputStreamReader (p. 1919)( System.in, false ), true );
```

See also

OutputStreamWriter (p. 2726)

Since

1.0

6.402.2 Constructor & Destructor Documentation

6.402.2.1 decaf::io::InputStreamReader::InputStreamReader (**InputStream** * stream, bool own = false)

Create a new **InputStreamReader** (p. 1919) that wraps the given **InputStream** (p. 1909).

Parameters

stream The **InputStream** (p. 1909) to read from. (cannot be null).

own Does this object own the passed **InputStream** (p. 1909) (defaults to false).

Exceptions

NullPointerException if the passed **InputStream** (p. 1909) is NULL.

6.402.2.2 `virtual decaf::io::InputStreamReader::~~InputStreamReader ()`
[virtual]

6.402.3 Member Function Documentation

6.402.3.1 `virtual void decaf::io::InputStreamReader::checkClosed () const throw (decaf::io::IOException)` [protected, virtual]

6.402.3.2 `virtual void decaf::io::InputStreamReader::close () throw (decaf::io::IOException)` [virtual]

6.402.3.3 `virtual int decaf::io::InputStreamReader::doReadArrayBounded (char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)` [protected, virtual]

Override this method to customize the functionality of the method `read(unsigned char* buffer, int size, int offset, int length)`.

All subclasses must override this method to provide the basic **Reader** (p. 2960) functionality.

Implements **decaf::io::Reader** (p. 2962).

6.402.3.4 `virtual bool decaf::io::InputStreamReader::ready () const throw (decaf::io::IOException)` [virtual]

Tells whether this stream is ready to be read.

Returns

True if the next `read()` (p. 2964) is guaranteed not to block for input, false otherwise. Note that returning false does not guarantee that the next read will block.

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented from **decaf::io::Reader** (p. 2965).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/InputStreamReader.h`

6.403 decaf::internal::nio::IntArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/IntArrayBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::IntArrayBuffer`:

Public Member Functions

- **IntArrayBuffer** (int size, bool readOnly=false) throw (decaf::lang::exceptions::IllegalArgumentException)

*Creates a **IntArrayBuffer** (p. 1921) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **IntArrayBuffer** (int *array, int size, int offset, int length, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

*Creates a **IntArrayBuffer** (p. 1921) object that wraps the given array.*

- **IntArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

- **IntArrayBuffer** (const IntArrayBuffer &other)

*Create a **IntArrayBuffer** (p. 1921) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual ~**IntArrayBuffer** ()
- virtual int * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

*the array that backs this **Buffer** (p. 855).*

Exceptions

***ReadOnlyBufferException** (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.*

- virtual int **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

***ReadOnlyBufferException** (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.*

- virtual IntBuffer * **asReadOnlyBuffer** () const

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only int buffer which the caller then owns.

- virtual `IntBuffer & compact ()` throw (`decaf::nio::ReadOnlyBufferException`)

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

*a reference to this **IntBuffer** (p. 1931)*

Exceptions

***ReadOnlyBufferException** (p. 2966) if this buffer is read-only.*

- virtual `IntBuffer * duplicate ()`

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*a new int **Buffer** (p. 855) which the caller owns.*

- virtual `int get ()` throw (`decaf::nio::BufferUnderflowException`)

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the int at the current position.

Exceptions

***BufferUnderflowException** (p. 882) if there no more data to return.*

- virtual `int get (int index) const` throw (`lang::exceptions::IndexOutOfBoundsException`)

Absolute get method.

Reads the value at the given index.

Parameters

***index** The index in the **Buffer** (p. 855) where the int is to be read.*

Returns

the int that is located at the given index.

Exceptions

***IndexOutOfBoundsException** if index is not smaller than the buffer's limit, or index is negative.*

- virtual `bool hasArray () const`

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual bool **isReadOnly** () const

- virtual IntBuffer & **put** (int value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes the given integer into this buffer at the current position, and then increments the position.

Parameters

value The integer value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.
ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual IntBuffer & **put** (int index, int value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes the given ints into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.
value The ints to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
ReadOnlyBufferException (p. 2966) - If this buffer is read-only.

- virtual IntBuffer * **slice** () const

*Creates a new **IntBuffer** (p. 1931) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the newly create **IntBuffer** (p. 1931) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

*Sets this **IntArrayBuffer** (p. 1921) as Read-Only.*

6.403.1 Constructor & Destructor Documentation

6.403.1.1 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (int size, bool readOnly = false) throw (decaf::lang::exceptions::IllegalArgumentException)`

Creates a **IntArrayBuffer** (p. 1921) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

size The size of the array, this is the limit we read and write to.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArgumentException if the capacity value is negative.

6.403.1.2 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (int * array, int size, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a **IntArrayBuffer** (p. 1921) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The actual array to wrap.

size The size of the given array.

offset The position that is this buffers start position.

length The limit of how many bytes into the array this Buffer can write.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException if buffer is NULL

IndexOutOfBoundsException if offset is greater than array capacity.

6.403.1.3 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **IntArrayBuffer** (p. 1921) will be that of the remaining capacity of the passed buffer.

Parameters

- array* The ByteArrayAdapter to wrap.
- offset* The position that is this buffers start position.
- length* The limit of how many bytes into the array this Buffer can write.
- readOnly* Boolean indicating if this buffer should be read-only, default as false.

Exceptions

- NullPointerException* if array is NULL
- IndexOutOfBoundsException* if offset + length is greater than array size.

6.403.1.4 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (const IntArrayBuffer & other)

Create a **IntArrayBuffer** (p.1921) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

Parameters

- other* The **IntArrayBuffer** (p.1921) this one is to mirror.

6.403.1.5 virtual decaf::internal::nio::IntArrayBuffer::~~IntArrayBuffer () [virtual]

6.403.2 Member Function Documentation

6.403.2.1 virtual int* decaf::internal::nio::IntArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

- the array that backs this **Buffer** (p. 855).

Exceptions

- ReadOnlyBufferException* (p. 2966) if this **Buffer** (p. 855) is read only.
- UnsupportedOperationException* if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p.1934).

6.403.2.2 `virtual int decaf::internal::nio::IntArrayBuffer::arrayOffset ()
throw (decaf::lang::exceptions::UnsupportedOperationException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 1934).

6.403.2.3 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::asReadOnlyBuffer
() const [virtual]`

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only int buffer which the caller then owns.

Implements **decaf::nio::IntBuffer** (p. 1934).

6.403.2.4 `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::compact ()
throw (decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 860) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 859) - 1 is copied to index `n = limit()` (p. 859) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **IntBuffer** (p. 1931)

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 1935).

6.403.2.5 virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::duplicate ()
[virtual]

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new int **Buffer** (p. 855) which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 1935).

6.403.2.6 virtual int decaf::internal::nio::IntArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the int at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return.

Implements **decaf::nio::IntBuffer** (p. 1937).

6.403.2.7 virtual int decaf::internal::nio::IntArrayBuffer::get (int index) const
throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the int is to be read.

Returns

the int that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::IntBuffer** (p. 1936).

6.403.2.8 **virtual bool decaf::internal::nio::IntArrayBuffer::hasArray () const**
[inline, virtual]

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::IntBuffer** (p. 1937).

6.403.2.9 **virtual bool decaf::internal::nio::IntArrayBuffer::isReadOnly () const**
[inline, virtual]

6.403.2.10 **virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put (int *index*,
int *value*) throw (lang::exceptions::IndexOutOfBoundsException,
decaf::nio::ReadOnlyBufferException)** [virtual]

Writes the given ints into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The ints to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written, or the index is negative.

ReadOnlyBufferException (p. 2966) - If this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 1938).

6.403.2.11 `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put
 (int value) throw (decaf::nio::BufferOverflowException,
 decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given integer into this buffer at the current position, and then increments the position.

Parameters

value The integer value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 1938).

6.403.2.12 `virtual void decaf::internal::nio::IntArrayBuffer::setReadOnly (bool
 value) [inline, protected, virtual]`

Sets this **IntArrayBuffer** (p. 1921) as Read-Only.

Parameters

value Boolean value, true if this buffer is to be read-only, false otherwise.

6.403.2.13 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::slice () const
 [virtual]`

Creates a new **IntBuffer** (p. 1931) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **IntBuffer** (p. 1931) which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 1940).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/IntArrayBuffer.h`

6.404 decaf::nio::IntBuffer Class Reference

This class defines four categories of operations upon int buffers:

```
#include <src/main/decaf/nio/IntBuffer.h>
```

Inheritance diagram for decaf::nio::IntBuffer:

Public Member Functions

- virtual **~IntBuffer** ()
- virtual std::string **toString** () const
- virtual int * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the int array that backs this buffer (optional operation).
- virtual int **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **IntBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only int buffer that shares this buffer's content.
- virtual **IntBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **IntBuffer** * **duplicate** ()=0
Creates a new int buffer that shares this buffer's content.
- virtual int **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual int **get** (int index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **IntBuffer** & **get** (std::vector< int > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **IntBuffer** & **get** (int *buffer, int size, int offset, int length) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible int array.
- **IntBuffer** & **put** (**IntBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException)

This method transfers the ints remaining in the given source buffer into this buffer.

- **IntBuffer** & **put** (const int *buffer, int size, int offset, int length) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

This method transfers ints into this buffer from the given source array.

- **IntBuffer** & **put** (std::vector< int > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source ints array into this buffer.

- virtual **IntBuffer** & **put** (int value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

Writes the given integer into this buffer at the current position, and then increments the position.

- virtual **IntBuffer** & **put** (int index, int value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

Writes the given ints into this buffer at the given index.

- virtual **IntBuffer** * **slice** () const =0

*Creates a new **IntBuffer** (p. 1931) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **IntBuffer** &value) const

- virtual bool **equals** (const **IntBuffer** &value) const

- virtual bool **operator==** (const **IntBuffer** &value) const

- virtual bool **operator<** (const **IntBuffer** &value) const

Static Public Member Functions

- static **IntBuffer** * **allocate** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)

Allocates a new Double buffer.

- static **IntBuffer** * **wrap** (int *array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

*Wraps the passed buffer with a new **IntBuffer** (p. 1931).*

- static **IntBuffer** * **wrap** (std::vector< int > &buffer)

*Wraps the passed STL int Vector in a **IntBuffer** (p. 1931).*

Protected Member Functions

- **IntBuffer** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)

*Creates a **IntBuffer** (p. 1931) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.404.1 Detailed Description

This class defines four categories of operations upon int buffers: o Absolute and relative get and put methods that read and write single ints; o Relative bulk get methods that transfer contiguous sequences of ints from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of ints from a int array or some other int buffer into this buffer o Methods for compacting, duplicating, and slicing a int buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing int array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.404.2 Constructor & Destructor Documentation

6.404.2.1 decaf::nio::IntBuffer::IntBuffer (int *capacity*) throw (decaf::lang::exceptions::IllegalArgumentException) [protected]

Creates a **IntBuffer** (p. 1931) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity The size and limit of the **Buffer** (p. 855) in integers.

Exceptions

IllegalArgumentException if capacity is negative.

6.404.2.2 virtual decaf::nio::IntBuffer::~~IntBuffer () [inline, virtual]

6.404.3 Member Function Documentation

6.404.3.1 static IntBuffer* decaf::nio::IntBuffer::allocate (int *capacity*) throw (decaf::lang::exceptions::IllegalArgumentException) [static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity The size of the Double buffer in integers.

Returns

the **IntBuffer** (p. 1931) that was allocated, caller owns.

6.404.3.2 `virtual int* decaf::nio::IntBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 855).

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1926).

6.404.3.3 `virtual int decaf::nio::IntBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1927).

6.404.3.4 `virtual IntBuffer* decaf::nio::IntBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow

the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only int buffer which the caller then owns.

Implemented in `decaf::internal::nio::IntArrayBuffer` (p. 1927).

6.404.3.5 `virtual IntBuffer& decaf::nio::IntBuffer::compact () throw (ReadOnlyBufferException)` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 860) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 859) - 1 is copied to index `n = limit()` (p. 859) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this `IntBuffer` (p. 1931)

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in `decaf::internal::nio::IntArrayBuffer` (p. 1927).

6.404.3.6 `virtual int decaf::nio::IntBuffer::compareTo (const IntBuffer & value) const` [virtual]

6.404.3.7 `virtual IntBuffer* decaf::nio::IntBuffer::duplicate ()` [pure virtual]

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new int `Buffer` (p. 855) which the caller owns.

Implemented in `decaf::internal::nio::IntArrayBuffer` (p. 1928).

6.404.3.8 `virtual bool decaf::nio::IntBuffer::equals (const IntBuffer & value)
const [virtual]`

6.404.3.9 `IntBuffer& decaf::nio::IntBuffer::get (std::vector< int > buffer) throw
(BufferUnderflowException)`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than length ints remaining in this buffer.

6.404.3.10 `virtual int decaf::nio::IntBuffer::get (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Absolute get method.

Reads the value at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the int is to be read.

Returns

the int that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1928).

6.404.3.11 `IntBuffer& decaf::nio::IntBuffer::get (int * buffer, int size,
int offset, int length) throw (BufferUnderflowException,
decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::NullPointerException)`

Relative bulk get method.

This method transfers ints from this buffer into the given destination array. If there are fewer ints remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 860), then no bytes are transferred and a **BufferUnderflowException** (p. 882) is thrown.

Otherwise, this method copies `length` ints from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters

buffer The pointer to an allocated buffer to fill.
size The size of the buffer that was passed in.
offset The position in the buffer to start filling.
length The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than `length` ints remaining in this buffer.
NullPointerException if the passed buffer is null.
IndexOutOfBoundsException if the preconditions of `size`, `offset`, or `length` are not met.

6.404.3.12 `virtual int decaf::nio::IntBuffer::get () throw (BufferUnderflowException) [pure virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the int at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1928).

6.404.3.13 `virtual bool decaf::nio::IntBuffer::hasArray () const [pure virtual]`

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1929).

6.404.3.14 `virtual bool decaf::nio::IntBuffer::operator< (const IntBuffer & value) const` [virtual]

6.404.3.15 `virtual bool decaf::nio::IntBuffer::operator== (const IntBuffer & value) const` [virtual]

6.404.3.16 `virtual IntBuffer& decaf::nio::IntBuffer::put (int value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given integer into this buffer at the current position, and then increments the position.

Parameters

value The integer value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in `decaf::internal::nio::IntArrayBuffer` (p. 1930).

6.404.3.17 `virtual IntBuffer& decaf::nio::IntBuffer::put (int index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given ints into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The ints to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written, or the index is negative.

ReadOnlyBufferException (p. 2966) - If this buffer is read-only.

Implemented in `decaf::internal::nio::IntArrayBuffer` (p. 1929).

6.404.3.18 `IntBuffer& decaf::nio::IntBuffer::put (const int * buffer, int size, int offset, int length) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)`

This method transfers ints into this buffer from the given source array.

If there are more ints to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 860), then no ints are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

- buffer* The array from which integers are to be read.
- size* The size of the buffer passed.
- offset* The offset within the array of the first char to be read.
- length* The number of integers to be read from the given array.

Returns

a reference to this buffer.

Exceptions

- BufferOverflowException** (p. 880) if there is insufficient space in this buffer.
- ReadOnlyBufferException** (p. 2966) if this buffer is read-only.
- NullPointerException** if the passed buffer is null.
- IndexOutOfBoundsException** if the preconditions of size, offset, or length are not met.

6.404.3.19 `IntBuffer& decaf::nio::IntBuffer::put (std::vector< int > & buffer) throw (BufferOverflowException, ReadOnlyBufferException)`

This method transfers the entire content of the given source ints array into this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size()`.

Parameters

- buffer* The buffer whose contents are copied to this **IntBuffer** (p. 1931).

Returns

a reference to this buffer.

Exceptions

- BufferOverflowException** (p. 880) if there is insufficient space in this buffer.
- ReadOnlyBufferException** (p. 2966) if this buffer is read-only.

6.404.3.20 `IntBuffer& decaf::nio::IntBuffer::put (IntBuffer & src) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException)`

This method transfers the ints remaining in the given source buffer into this buffer.

If there are more ints remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 860), then no ints are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies `n = src.remaining()` ints from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

src The buffer to take ints from an place in this one.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer for the remaining ints in the source buffer.

IllegalArgumentException if the source buffer is this buffer.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

6.404.3.21 virtual IntBuffer* decaf::nio::IntBuffer::slice () const [pure virtual]

Creates a new **IntBuffer** (p. 1931) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **IntBuffer** (p. 1931) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1930).

6.404.3.22 virtual std::string decaf::nio::IntBuffer::toString () const [virtual]

Returns

a `std::string` describing this object

6.404.3.23 static IntBuffer* decaf::nio::IntBuffer::wrap (int * array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [static]

Wraps the passed buffer with a new **IntBuffer** (p. 1931).

The new buffer will be backed by the given int array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

- array*** The array that will back the new buffer.
- size*** The size of the passed in array.
- offset*** The offset of the subarray to be used.
- length*** The length of the subarray to be used.

Returns

a new **IntBuffer** (p.1931) that is backed by `buffer`, caller owns.

Exceptions

- NullPointerException*** if the array pointer is NULL.
- IndexOutOfBoundsException*** if the preconditions of size, offset, or length are not met.

6.404.3.24 `static IntBuffer* decaf::nio::IntBuffer::wrap (std::vector< int > &
buffer) [static]`

Wraps the passed STL int Vector in a **IntBuffer** (p.1931).

The new buffer will be backed by the given int array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

- buffer*** The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new **IntBuffer** (p.1931) that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/IntBuffer.h`

6.405 decaf::lang::Integer Class Reference

```
#include <src/main/decaf/lang/Integer.h>
```

Inheritance diagram for `decaf::lang::Integer`:

Public Member Functions

- **Integer** (int value)
- **Integer** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Integer** ()
- virtual int **compareTo** (const **Integer** &i) const
*Compares this **Integer** (p. 1941) instance with another.*
- bool **equals** (const **Integer** &i) const
- virtual bool **operator==** (const **Integer** &i) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Integer** &i) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const int &i) const
*Compares this **Integer** (p. 1941) instance with another.*
- bool **equals** (const int &i) const
- virtual bool **operator==** (const int &i) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const int &i) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static **Integer** **decode** (const std::string &value) throw (exceptions::NumberFormatException)
*Decodes a **String** (p. 3427) into a **Integer** (p. 1941).*
- static int **reverseBytes** (int value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.
- static int **reverse** (int value)
Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.
- static int **parseInt** (const std::string &s, int radix) throw (exceptions::NumberFormatException)
Parses the string argument as a signed int in the radix specified by the second argument.
- static int **parseInt** (const std::string &s) throw (exceptions::NumberFormatException)
Parses the string argument as a signed decimal int.
- static **Integer** **valueOf** (int value)
*Returns a **Integer** (p. 1941) instance representing the specified int value.*
- static **Integer** **valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Integer** (p. 1941) object holding the value given by the specified std::string.*
- static **Integer** **valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Integer** (p. 1941) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*
- static int **bitCount** (int value)
Returns the number of one-bits in the two's complement binary representation of the specified int value.
- static std::string **toString** (int value)
*Converts the int to a **String** (p. 3427) representation.*
- static std::string **toString** (int value, int radix)
Returns a string representation of the first argument in the radix specified by the second argument.
- static std::string **toHexString** (int value)
Returns a string representation of the integer argument as an unsigned integer in base 16.
- static std::string **toOctalString** (int value)
Returns a string representation of the integer argument as an unsigned integer in base 8.
- static std::string **toBinaryString** (int value)
Returns a string representation of the integer argument as an unsigned integer in base 2.

- static int **highestOneBit** (int value)
Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.
- static int **lowestOneBit** (int value)
Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.
- static int **numberOfLeadingZeros** (int value)
Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.
- static int **numberOfTrailingZeros** (int value)
Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.
- static int **rotateLeft** (int value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.
- static int **rotateRight** (int value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.
- static int **signum** (int value)
Returns the signum function of the specified int value.

Static Public Attributes

- static const int **SIZE** = 32
The size in bits of the primitive int type.
- static const int **MAX_VALUE** = (int)0x7FFFFFFF
The maximum value that the primitive type can hold.
- static const int **MIN_VALUE** = (int)0x80000000
The minimum value that the primitive type can hold.

6.405.1 Constructor & Destructor Documentation

6.405.1.1 decaf::lang::Integer::Integer (int value)

Parameters

value The primitive value to wrap in an Integer (p. 1941) instance.

6.405.1.2 `decaf::lang::Integer::Integer (const std::string & value) throw (exceptions::NumberFormatException)`

Parameters

value The base 10 encoded string to decode to an `Integer` (p.1941) and wrap.

Exceptions

NumberFormatException

6.405.1.3 `virtual decaf::lang::Integer::~~Integer () [inline, virtual]`

6.405.2 Member Function Documentation

6.405.2.1 `static int decaf::lang::Integer::bitCount (int value) [static]`

Returns the number of one-bits in the two's complement binary representation of the specified int value.

This function is sometimes referred to as the population count.

Parameters

value - the int to count

Returns

the number of one-bits in the two's complement binary representation of the specified int value.

6.405.2.2 `virtual unsigned char decaf::lang::Integer::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

6.405.2.3 `virtual int decaf::lang::Integer::compareTo (const int & i) const [virtual]`

Compares this `Integer` (p.1941) instance with another.

Parameters

i - the `Integer` (p.1941) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< int >` (p.1126).

6.405.2.4 `virtual int decaf::lang::Integer::compareTo (const Integer & i) const`
[virtual]

Compares this **Integer** (p.1941) instance with another.

Parameters

i - the **Integer** (p.1941) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.405.2.5 `static Integer decaf::lang::Integer::decode (const std::string & value)`
`throw (exceptions::NumberFormatException)` [static]

Decodes a **String** (p.3427) into a **Integer** (p.1941).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified **String** (p.3427) is the minus sign. No whitespace characters are permitted in the string.

Parameters

value - The string to decode

Returns

a **Integer** (p.1941) object containing the decoded value

Exceptions

NumberFormatException if the string is not formatted correctly.

6.405.2.6 `virtual double decaf::lang::Integer::doubleValue () const` [inline,
virtual]

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.405.2.7 `bool decaf::lang::Integer::equals (const Integer & i) const` [inline]**Parameters**

i - the **Integer** (p.1941) object to compare against.

Returns

true if the two **Integer** (p.1941) Objects have the same value.

6.405.2.8 `bool decaf::lang::Integer::equals (const int & i) const [inline, virtual]`

Parameters

i - the **Integer** (p.1941) object to compare against.

Returns

true if the two **Integer** (p.1941) Objects have the same value.

Implements **decaf::lang::Comparable**< **int** > (p.1126).

6.405.2.9 `virtual float decaf::lang::Integer::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.405.2.10 `static int decaf::lang::Integer::highestOneBit (int value) [static]`

Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

value - the int to be inspected

Returns

an int value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.405.2.11 `virtual int decaf::lang::Integer::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.405.2.12 `virtual long long decaf::lang::Integer::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.405.2.13 `static int decaf::lang::Integer::lowestOneBit (int value) [static]`

Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

value - the int to be inspected

Returns

an int value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.405.2.14 `static int decaf::lang::Integer::numberOfLeadingZeros (int value) [static]`

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.

Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values x:

$\ast \text{floor}(\log_2(x)) = 31 - \text{numberOfLeadingZeros}(x) \ast \text{ceil}(\log_2(x)) = 32 - \text{numberOfLeadingZeros}(x - 1)$

Parameters

value - the int to be inspected

Returns

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.405.2.15 `static int decaf::lang::Integer::numberOfTrailingZeros (int value) [static]`

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.

Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Parameters

value - the int to be inspected

Returns

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.405.2.16 `virtual bool decaf::lang::Integer::operator< (const int & i) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

i - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< int >` (p.1127).

6.405.2.17 `virtual bool decaf::lang::Integer::operator< (const Integer & i) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

i - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.405.2.18 `virtual bool decaf::lang::Integer::operator== (const Integer & i)`
`const` [inline, virtual]

Compares equality between this object and the one passed.

Parameters

i - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.405.2.19 `virtual bool decaf::lang::Integer::operator==(const int & i) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

i - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< int > (p.1127).

6.405.2.20 `static int decaf::lang::Integer::parseInt (const std::string & s, int`
radix) throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed int in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p.1022) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type **NumberFormatException** is thrown if any of the following situations occurs:
* The first argument is null or is a string of length zero. * The radix is either smaller than **Character.MIN_RADIX** (p.1026) or larger than **Character.MAX_RADIX** (p.1026). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type int.

Parameters

s - the **String** (p.3427) containing the int representation to be parsed

radix - the radix to be used while parsing *s*

Returns

the int represented by the string argument in the specified radix.

Exceptions

NumberFormatException - If **String** (p.3427) does not contain a parsable int.

6.405.2.21 `static int decaf::lang::Integer::parseInt (const std::string & s) throw`
(exceptions::NumberFormatException) [static]

Parses the string argument as a signed decimal int.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting int value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseInt(const std::string, int)` method.

Parameters

s - **String** (p. 3427) to convert to a int

Returns

the converted int value

Exceptions

NumberFormatException if the string is not a int.

6.405.2.22 static int decaf::lang::Integer::reverse (int *value*) [static]

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.

Parameters

value - the value whose bits are to be reversed

Returns

the reversed bits int.

6.405.2.23 static int decaf::lang::Integer::reverseBytes (int *value*) [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.

Parameters

value - the int whose bytes we are to reverse

Returns

the reversed int.

6.405.2.24 static int decaf::lang::Integer::rotateLeft (int *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

(Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: rotateLeft(val, -distance) == rotateRight(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F).

Parameters

value - the int to be inspected

distance - the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

6.405.2.25 `static int decaf::lang::Integer::rotateRight (int value, int distance)`
[static]

Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

(Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: `rotateRight(val, -distance) == rotateLeft(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateRight(val, distance) == rotateRight(val, distance & 0x1F)`.

Parameters

value - the int to be inspected

distance - the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

6.405.2.26 `virtual short decaf::lang::Integer::shortValue () const` [inline, virtual]

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

6.405.2.27 `static int decaf::lang::Integer::signum (int value)` [static]

Returns the signum function of the specified int value.

(The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

Parameters

value - the int to be inspected

Returns

the signum function of the specified int value.

6.405.2.28 `static std::string decaf::lang::Integer::toBinaryString (int value)`
[static]

Returns a string representation of the integer argument as an unsigned integer in base 2.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character.

The characters '0' and '1' are used as binary digits.

Parameters

value - the int to be translated to a binary string

Returns

the unsigned int value as a binary string

6.405.2.29 `static std::string decaf::lang::Integer::toHexString (int value)`
[static]

Returns a string representation of the integer argument as an unsigned integer in base 16.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

Parameters

value - the int to be translated to an Octal string

Returns

the unsigned int value as a Octal string

6.405.2.30 `static std::string decaf::lang::Integer::toOctalString (int value)`
[static]

Returns a string representation of the integer argument as an unsigned integer in base 8.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

Parameters

value - the int to be translated to an Octal string

Returns

the unsigned int value as a Octal string

6.405.2.31 `std::string decaf::lang::Integer::toString () const`**Returns**

this Integer (p. 1941) Object as a **String** (p. 3427) Representation

6.405.2.32 `static std::string decaf::lang::Integer::toString (int value, int radix) [static]`

Returns a string representation of the first argument in the radix specified by the second argument.

If the radix is smaller than **Character.MIN_RADIX** (p. 1026) or larger than **Character.MAX_RADIX** (p. 1026), then the radix 10 is used instead.

If the first argument is negative, the first element of the result is the ASCII minus character '-'. If the first argument is not negative, no sign character appears in the result.

The remaining characters of the result represent the magnitude of the first argument. If the magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the magnitude will not be the zero character. The following ASCII characters are used as digits:

0123456789abcdefghijklmnopqrstuvwxyz

Parameters

value - the int to convert to a string

radix - the radix to format the string in

Returns

an int formatted to the string value of the radix given.

6.405.2.33 `static std::string decaf::lang::Integer::toString (int value) [static]`

Converts the int to a **String** (p. 3427) representation.

Parameters

value The int to convert to a `std::string` instance.

Returns

string representation

6.405.2.34 `static Integer decaf::lang::Integer::valueOf (const std::string & value,
int radix) throw (exceptions::NumberFormatException) [static]`

Returns a **Integer** (p. 1941) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed int in the radix specified by the second argument, exactly as if the argument were given to the `parseInt(std::string, int)` method. The result is a **Integer** (p. 1941) object that represents the int value specified by the string.

Parameters

value - `std::string` to parse as base (*radix*)

radix - base of the string to parse.

Returns

new **Integer** (p. 1941) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a valid int.

6.405.2.35 `static Integer decaf::lang::Integer::valueOf (int value) [inline,
static]`

Returns a **Integer** (p. 1941) instance representing the specified int value.

Parameters

value - the int to wrap

Returns

the new **Integer** (p. 1941) object wrapping value.

6.405.2.36 `static Integer decaf::lang::Integer::valueOf (const std::string & value
) throw (exceptions::NumberFormatException) [static]`

Returns a **Integer** (p. 1941) object holding the value given by the specified `std::string`.

The argument is interpreted as representing a signed decimal int, exactly as if the argument were given to the `parseInt(std::string)` method. The result is a **Integer** (p. 1941) object that represents the int value specified by the string.

Parameters

value - `std::string` to parse as base 10

Returns

new **Integer** (p. 1941) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a decimal int.

6.405.3 Field Documentation

6.405.3.1 `const int decaf::lang::Integer::MAX_VALUE = (int)0x7FFFFFFF` [static]

The maximum value that the primitive type can hold.

6.405.3.2 `const int decaf::lang::Integer::MIN_VALUE = (int)0x80000000` [static]

The minimum value that the primitive type can hold.

6.405.3.3 `const int decaf::lang::Integer::SIZE = 32` [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Integer.h`

6.406 activemq::commands::IntegerResponse Class Reference

```
#include <src/main/activemq/commands/IntegerResponse.h>
```

Inheritance diagram for `activemq::commands::IntegerResponse`:

Public Member Functions

- `IntegerResponse ()`
- `virtual ~IntegerResponse ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual IntegerResponse * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- `virtual void copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this object's members, overwriting any existing data.
- `virtual std::string toString () const`
Returns a string containing the information for this DataStructure (p. 1553) such as its type and value of its elements.
- `virtual bool equals (const DataStructure *value) const`
Compares the DataStructure (p. 1553) passed in to this one, and returns if they are equivalent.
- `virtual int getResult () const`
- `virtual void setResult (int result)`

Static Public Attributes

- static const unsigned char **ID_INTEGERRESPONSE** = 34

Protected Attributes

- int **result**

6.406.1 Constructor & Destructor Documentation

6.406.1.1 `activemq::commands::IntegerResponse::IntegerResponse ()`

6.406.1.2 `virtual activemq::commands::IntegerResponse::~~IntegerResponse ()`
[virtual]

6.406.2 Member Function Documentation

6.406.2.1 `virtual IntegerResponse* activemq::commands::IntegerResponse::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p.3077).

6.406.2.2 `virtual void activemq::commands::IntegerResponse::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::Response` (p.3078).

6.406.2.3 `virtual bool activemq::commands::IntegerResponse::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p.1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Response` (p.3078).

6.406.2.4 `virtual unsigned char activemq::commands::IntegerResponse::getDataStructureType () const`
 [virtual]

Get the unique identifier that this object and its own Marshaller share.

Returns

new **DataSet** (p. 1553) type copy.

Reimplemented from **activemq::commands::Response** (p. 3078).

6.406.2.5 `virtual int activemq::commands::IntegerResponse::getResult () const`
 [virtual]

6.406.2.6 `virtual void activemq::commands::IntegerResponse::setResult (int result)` [virtual]

6.406.2.7 `virtual std::string activemq::commands::IntegerResponse::toString () const` [virtual]

Returns a string containing the information for this **DataSet** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3079).

6.406.3 Field Documentation

6.406.3.1 `const unsigned char activemq::commands::IntegerResponse::ID _ - INTEGERRESPONSE = 34` [static]

6.406.3.2 `int activemq::commands::IntegerResponse::result` [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**IntegerResponse.h**

6.407 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1958).

#include <src/main/activemq/wireformat/openwire/marshal/v6/IntegerResponseMarshaller.h>

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller**:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.407.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1958). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.407.2 Constructor & Destructor Documentation

6.407.2.1 `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

6.407.2.2 `virtual activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

6.407.3 Member Function Documentation

6.407.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3108).

6.407.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109).

6.407.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109).

6.407.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3109).

6.407.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3110).

6.407.3.6 virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3110).

6.407.3.7 virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3111).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**IntegerResponseMarshaller.h**

6.408 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1962).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller`:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.408.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1962). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.408.2 Constructor & Destructor Documentation

6.408.2.1 `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

6.408.2.2 `virtual activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

6.408.3 Member Function Documentation

6.408.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3090).

6.408.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3091).

6.408.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseMarshal(OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3091).

6.408.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3091).

6.408.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3092).

6.408.3.6 virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3092).

6.408.3.7 virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3093).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**IntegerResponseMarshaller.h**

6.409 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1966).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller`:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.409.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1966). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.409.2 Constructor & Destructor Documentation

6.409.2.1 `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

6.409.2.2 `virtual activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

6.409.3 Member Function Documentation

6.409.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3099).

6.409.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3100).

6.409.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3100).

6.409.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3100).

6.409.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3101).

6.409.3.6 virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3101).

6.409.3.7 virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3102).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**IntegerResponseMarshaller.h**

6.410 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1970).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller`:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.410.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1970). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.410.2 Constructor & Destructor Documentation

6.410.2.1 `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

6.410.2.2 `virtual activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

6.410.3 Member Function Documentation

6.410.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3086).

6.410.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3086).

6.410.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3086).

6.410.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3087).

6.410.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3087).

6.410.3.6 virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3088).

6.410.3.7 virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3088).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**IntegerResponseMarshaller.h**

6.411 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1974).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller`:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.411.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1974). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.411.2 Constructor & Destructor Documentation

6.411.2.1 `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::IntegerResponseMarshaller() [inline]`

6.411.2.2 `virtual activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::~~IntegerResponseMarshaller() [inline, virtual]`

6.411.3 Member Function Documentation

6.411.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::createObject() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3104).

6.411.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3104).

6.411.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3104).

6.411.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3105).

6.411.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3105).

6.411.3.6 virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3106).

6.411.3.7 virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3106).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**IntegerResponseMarshaller.h**

6.412 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1978).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller`:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.412.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1978). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.412.2 Constructor & Destructor Documentation

6.412.2.1 `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

6.412.2.2 `virtual activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

6.412.3 Member Function Documentation

6.412.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3095).

6.412.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3095).

6.412.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3095).

6.412.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3096).

6.412.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3096).

6.412.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3097).

6.412.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3097).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h`

6.413 `internal_state` Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

Data Fields

- `z_stream` strm
- `int` status
- `Bytef *` pending_buf
- `ulg` pending_buf_size
- `Bytef *` pending_out
- `uInt` pending
- `int` wrap
- `gz_headerp` gzhead
- `uInt` gzindex
- `Byte` method
- `int` last_flush
- `uInt` w_size
- `uInt` w_bits
- `uInt` w_mask
- `Bytef *` window
- `ulg` window_size
- `Posf *` prev
- `Posf *` head
- `uInt` ins_h
- `uInt` hash_size
- `uInt` hash_bits
- `uInt` hash_mask
- `uInt` hash_shift
- `long` block_start
- `uInt` match_length
- `IPos` prev_match
- `int` match_available
- `uInt` strstart
- `uInt` match_start
- `uInt` lookahead
- `uInt` prev_length
- `uInt` max_chain_length
- `uInt` max_lazy_match
- `int` level
- `int` strategy
- `uInt` good_match
- `int` nice_match
- `struct ct_data_s` dyn_ltree [HEAP_SIZE]
- `struct ct_data_s` dyn_dtree [2 * D_CODES + 1]
- `struct ct_data_s` bl_tree [2 * BL_CODES + 1]
- `struct tree_desc_s` l_desc
- `struct tree_desc_s` d_desc
- `struct tree_desc_s` bl_desc
- `ush` bl_count [MAX_BITS + 1]
- `int` heap [2 * L_CODES + 1]
- `int` heap_len
- `int` heap_max
- `uch` depth [2 * L_CODES + 1]

- `uchf * l_buf`
- `uInt lit_bufsize`
- `uInt last_lit`
- `ushf * d_buf`
- `ulg opt_len`
- `ulg static_len`
- `uInt matches`
- `int last_eob_len`
- `ush bi_buf`
- `int bi_valid`
- `ulg high_water`
- `int dummy`

6.413.1 Field Documentation

- 6.413.1.1 `ush internal_state::bi_buf`
- 6.413.1.2 `int internal_state::bi_valid`
- 6.413.1.3 `ush internal_state::bl_count[MAX_BITS+1]`
- 6.413.1.4 `struct tree_desc_s internal_state::bl_desc`
- 6.413.1.5 `struct ct_data_s internal_state::bl_tree[2 * BL_CODES+1]`
- 6.413.1.6 `long internal_state::block_start`
- 6.413.1.7 `ushf* internal_state::d_buf`
- 6.413.1.8 `struct tree_desc_s internal_state::d_desc`
- 6.413.1.9 `uch internal_state::depth[2 * L_CODES+1]`
- 6.413.1.10 `int internal_state::dummy`
- 6.413.1.11 `struct ct_data_s internal_state::dyn_dtree[2 * D_CODES+1]`
- 6.413.1.12 `struct ct_data_s internal_state::dyn_ltree[HEAP_SIZE]`
- 6.413.1.13 `uInt internal_state::good_match`
- 6.413.1.14 `gz_headerp internal_state::gzhead`
- 6.413.1.15 `uInt internal_state::gzindex`
- 6.413.1.16 `uInt internal_state::hash_bits`
- 6.413.1.17 `uInt internal_state::hash_mask`
- 6.413.1.18 `uInt internal_state::hash_shift`
- 6.413.1.19 `uInt internal_state::hash_size`
- 6.413.1.20 `Posf* internal_state::head`
- 6.413.1.21 `int internal_state::heap[2 * L_CODES+1]`
- 6.413.1.22 `int internal_state::heap_len`
- 6.413.1.23 `int internal_state::heap_max`
- 6.413.1.24 `ulg internal_state::high_water`
- 6.413.1.25 `uInt internal_state::ins_h`
- 6.413.1.26 `uchf* internal_state::l_buf`
- 6.413.1.27 `struct tree_desc_s internal_state::l_desc`
- 6.413.1.28 `int internal_state::last_eob_len`
- 6.413.1.29 `int internal_state::last_flush`
- 6.413.1.30 `uInt internal_state::last_lit`

- `src/main/decaf/internal/util/zip/deflate.h`
- `src/main/decaf/internal/util/zip/zlib.h`

6.414 activemq::transport::mock::InternalCommandListener Class Reference

Listens for Commands sent from the **MockTransport** (p. 2592).

```
#include <src/main/activemq/transport/mock/InternalCommandListener.h>
```

Inheritance diagram for `activemq::transport::mock::InternalCommandListener`:

Public Member Functions

- **InternalCommandListener** ()
- virtual **~InternalCommandListener** ()
- void **setTransport** (**MockTransport** *transport)
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
- void **run** ()

Default implementation of the run method - does nothing.

6.414.1 Detailed Description

Listens for Commands sent from the **MockTransport** (p. 2592). This class processes all outbound commands and sends responses that are constructed by calling the Protocol provided **ResponseBuilder** (p. 3079) and getting a set of Commands to send back into the **MockTransport** (p. 2592) as incoming Commands and Responses.

6.414.2 Constructor & Destructor Documentation

- 6.414.2.1 `activemq::transport::mock::InternalCommandListener::InternalCommandListener ()`
- 6.414.2.2 `virtual activemq::transport::mock::InternalCommandListener::~~InternalCommandListener () [virtual]`

6.414.3 Member Function Documentation

- 6.414.3.1 `virtual void activemq::transport::mock::InternalCommandListener::onCommand (const Pointer< Command > & command) [virtual]`
- 6.414.3.2 `void activemq::transport::mock::InternalCommandListener::run () [virtual]`

Default implementation of the run method - does nothing.

Reimplemented from `decaf::lang::Thread` (p. 3527).

- 6.414.3.3 `void activemq::transport::mock::InternalCommandListener::setResponseBuilder (const Pointer< ResponseBuilder > & responseBuilder) [inline]`
- 6.414.3.4 `void activemq::transport::mock::InternalCommandListener::setTransport (MockTransport * transport) [inline]`

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/InternalCommandListener.h`

6.415 decaf::lang::exceptions::InterruptedException Class Reference

```
#include <src/main/decaf/lang/exceptions/InterruptedException.h>
```

Inheritance diagram for `decaf::lang::exceptions::InterruptedException`:

Public Member Functions

- `InterruptedException () throw ()`
Default Constructor.
- `InterruptedException (const Exception &ex) throw ()`
*Conversion Constructor from some other **Exception** (p. 1712).*
- `InterruptedException (const InterruptedException &ex) throw ()`

Copy Constructor.

- **InterruptedException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- **InterruptedException** (const std::exception *cause) throw ()

Constructor.

- **InterruptedException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **InterruptedException** * clone () const

Clones this exception.

- virtual ~**InterruptedException** () throw ()

6.415.1 Constructor & Destructor Documentation

6.415.1.1 decaf::lang::exceptions::InterruptedException::InterruptedException () throw () [inline]

Default Constructor.

6.415.1.2 decaf::lang::exceptions::InterruptedException::InterruptedException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.415.1.3 decaf::lang::exceptions::InterruptedException::InterruptedException (const InterruptedException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.415.1.4 decaf::lang::exceptions::InterruptedException::InterruptedException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.415.1.5 `decaf::lang::exceptions::InterruptedException::InterruptedException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause **Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.415.1.6 `decaf::lang::exceptions::InterruptedException::InterruptedException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.415.1.7 `virtual decaf::lang::exceptions::InterruptedException::~InterruptedException () throw () [inline, virtual]`

6.415.2 Member Function Documentation

6.415.2.1 `virtual InterruptedException* decaf::lang::exceptions::InterruptedException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/InterruptedException.h`

6.416 decaf::io::InterruptedException Class Reference

```
#include <src/main/decaf/io/InterruptedException.h>
```

Inheritance diagram for `decaf::io::InterruptedException`:

Public Member Functions

- **InterruptedException** () throw ()
Default Constructor.
- **InterruptedException** (const **lang::Exception** &ex) throw ()
Copy Constructor.
- **InterruptedException** (const **InterruptedException** &ex) throw ()
Copy Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InterruptedException** (const std::exception *cause) throw ()
Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **InterruptedException** * clone () const
Clones this exception.
- virtual ~**InterruptedException** () throw ()

6.416.1 Constructor & Destructor Documentation

6.416.1.1 `decaf::io::InterruptedException::InterruptedException () throw () [inline]`

Default Constructor.

6.416.1.2 `decaf::io::InterruptedIOException::InterruptedIOException (const lang::Exception & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy

6.416.1.3 `decaf::io::InterruptedIOException::InterruptedIOException (const InterruptedIOException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.416.1.4 `decaf::io::InterruptedIOException::InterruptedIOException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.416.1.5 `decaf::io::InterruptedIOException::InterruptedIOException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.416.1.6 `decaf::io::InterruptedIOException::InterruptedIOException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.416.1.7 virtual decaf::io::InterruptedIOException::~InterruptedIOException () throw () [inline, virtual]

6.416.2 Member Function Documentation

6.416.2.1 virtual InterruptedIOException* decaf::io::InterruptedIOException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

- a new exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2005).

Reimplemented in **decaf::net::SocketTimeoutException** (p. 3321).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InterruptedIOException.h**

6.417 cms::InvalidClientIdException Class Reference

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

```
#include <src/main/cms/InvalidClientIdException.h>
```

Inheritance diagram for cms::InvalidClientIdException:

Public Member Functions

- **InvalidClientIdException** () throw ()
- **InvalidClientIdException** (const **InvalidClientIdException** &ex) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception *cause) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidClientIdException** () throw ()

6.417.1 Detailed Description

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

Since

1.3

6.417.2 Constructor & Destructor Documentation

6.417.2.1 `cms::InvalidClientIdException::InvalidClientIdException () throw ()`

6.417.2.2 `cms::InvalidClientIdException::InvalidClientIdException (const InvalidClientIdException & ex) throw ()`

6.417.2.3 `cms::InvalidClientIdException::InvalidClientIdException (const std::string & message, const std::exception * cause) throw ()`

6.417.2.4 `cms::InvalidClientIdException::InvalidClientIdException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`

6.417.2.5 `virtual cms::InvalidClientIdException::~InvalidClientIdException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/InvalidClientIdException.h`

6.418 cms::InvalidDestinationException Class Reference

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

```
#include <src/main/cms/InvalidDestinationException.h>
```

Inheritance diagram for cms::InvalidDestinationException:

Public Member Functions

- `InvalidDestinationException () throw ()`
- `InvalidDestinationException (const InvalidDestinationException &ex) throw ()`
- `InvalidDestinationException (const std::string &message, const std::exception *cause) throw ()`
- `InvalidDestinationException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~InvalidDestinationException () throw ()`

6.418.1 Detailed Description

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

Since

1.3

6.418.2 Constructor & Destructor Documentation

```
6.418.2.1 cms::InvalidDestinationException::InvalidDestinationException ( )
throw ()
```

```
6.418.2.2 cms::InvalidDestinationException::InvalidDestinationException ( const
InvalidDestinationException & ex ) throw ()
```

```
6.418.2.3 cms::InvalidDestinationException::InvalidDestinationException ( const
std::string & message, const std::exception * cause ) throw ()
```

```
6.418.2.4 cms::InvalidDestinationException::InvalidDestinationException ( const
std::string & message, const std::exception * cause, const std::vector<
std::pair< std::string, int > > & stackTrace ) throw ()
```

```
6.418.2.5 virtual cms::InvalidDestinationException::~InvalidDestinationException
( ) throw () [virtual]
```

The documentation for this class was generated from the following file:

- `src/main/cms/InvalidDestinationException.h`

6.419 decaf::security::InvalidKeyException Class Reference

```
#include <src/main/decaf/security/InvalidKeyException.h>
```

Inheritance diagram for decaf::security::InvalidKeyException:

Public Member Functions

- **InvalidKeyException** () throw ()
Default Constructor.
- **InvalidKeyException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **InvalidKeyException** (const **InvalidKeyException** &ex) throw ()
Copy Constructor.

- **InvalidKeyException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InvalidKeyException** (const std::exception *cause) throw ()
Constructor.
- **InvalidKeyException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InvalidKeyException * clone** () const
Clones this exception.
- virtual **~InvalidKeyException** () throw ()

6.419.1 Constructor & Destructor Documentation

6.419.1.1 `decaf::security::InvalidKeyException::InvalidKeyException () throw ()` [inline]

Default Constructor.

6.419.1.2 `decaf::security::InvalidKeyException::InvalidKeyException (const Exception & ex) throw ()` [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.419.1.3 `decaf::security::InvalidKeyException::InvalidKeyException (const InvalidKeyException & ex) throw ()` [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.419.1.4 `decaf::security::InvalidKeyException::InvalidKeyException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.419.1.5 `decaf::security::InvalidKeyException::InvalidKeyException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.419.1.6 `decaf::security::InvalidKeyException::InvalidKeyException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs
lineNumber line number where the exception occurred.
msg message to report
... list of primitives that are formatted into the message

6.419.1.7 `virtual decaf::security::InvalidKeyException::~~InvalidKeyException () throw () [inline, virtual]`

6.419.2 Member Function Documentation

6.419.2.1 `virtual InvalidKeyException* decaf::security::InvalidKeyException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::KeyException** (p. 2153).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/InvalidKeyException.h`

6.420 decaf::nio::InvalidMarkException Class Reference

```
#include <src/main/decaf/nio/InvalidMarkException.h>
```

Inheritance diagram for `decaf::nio::InvalidMarkException`:

Public Member Functions

- **InvalidMarkException** () throw ()
Default Constructor.
- **InvalidMarkException** (const **lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **InvalidMarkException** (const **InvalidMarkException** &ex) throw ()
Copy Constructor.
- **InvalidMarkException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InvalidMarkException** (const std::exception *cause) throw ()
Constructor.
- **InvalidMarkException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InvalidMarkException** * clone () const
Clones this exception.
- virtual ~**InvalidMarkException** () throw ()

6.420.1 Constructor & Destructor Documentation

6.420.1.1 decaf::nio::InvalidMarkException::InvalidMarkException () throw ()
[inline]

Default Constructor.

6.420.1.2 decaf::nio::InvalidMarkException::InvalidMarkException (const lang::Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex The Exception whose state data is to be copied into this Exception.

6.420.1.3 decaf::nio::InvalidMarkException::InvalidMarkException (const InvalidMarkException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex The Exception whose state data is to be copied into this Exception.

6.420.1.4 decaf::nio::InvalidMarkException::InvalidMarkException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.420.1.5 decaf::nio::InvalidMarkException::InvalidMarkException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.420.1.6 decaf::nio::InvalidMarkException::InvalidMarkException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.420.1.7 `virtual decaf::nio::InvalidMarkException::~InvalidMarkException ()
throw () [inline, virtual]`

6.420.2 Member Function Documentation

6.420.2.1 `virtual InvalidMarkException* decaf::nio::InvalidMarkException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new Exception instance that is a copy of this Exception.

Reimplemented from `decaf::lang::exceptions::IllegalStateException` (p. 1871).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/InvalidMarkException.h`

6.421 cms::InvalidSelectorException Class Reference

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

```
#include <src/main/cms/InvalidSelectorException.h>
```

Inheritance diagram for `cms::InvalidSelectorException`:

Public Member Functions

- `InvalidSelectorException () throw ()`
- `InvalidSelectorException (const InvalidSelectorException &ex) throw ()`
- `InvalidSelectorException (const std::string &message, const std::exception *cause) throw ()`
- `InvalidSelectorException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~InvalidSelectorException () throw ()`

6.421.1 Detailed Description

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

Since

1.3

6.421.2 Constructor & Destructor Documentation

6.421.2.1 `cms::InvalidSelectorException::InvalidSelectorException () throw ()`

6.421.2.2 `cms::InvalidSelectorException::InvalidSelectorException (const InvalidSelectorException & ex) throw ()`

6.421.2.3 `cms::InvalidSelectorException::InvalidSelectorException (const std::string & message, const std::exception * cause) throw ()`

6.421.2.4 `cms::InvalidSelectorException::InvalidSelectorException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`

6.421.2.5 `virtual cms::InvalidSelectorException::~InvalidSelectorException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/InvalidSelectorException.h`

6.422 decaf::lang::exceptions::InvalidStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/InvalidStateException.h>
```

Inheritance diagram for decaf::lang::exceptions::InvalidStateException:

Public Member Functions

- `InvalidStateException () throw ()`

Default Constructor.

- `InvalidStateException (const Exception &ex) throw ()`

*Conversion Constructor from some other **Exception** (p. 1712).*

- `InvalidStateException (const InvalidStateException &ex) throw ()`

Copy Constructor.

- **InvalidStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InvalidStateException** (const std::exception *cause) throw ()
Constructor.
- **InvalidStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InvalidStateException * clone** () const
Clones this exception.
- virtual **~InvalidStateException** () throw ()

6.422.1 Constructor & Destructor Documentation

6.422.1.1 decaf::lang::exceptions::InvalidStateException::InvalidStateException () throw () [inline]

Default Constructor.

6.422.1.2 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.422.1.3 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const InvalidStateException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.422.1.4 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.422.1.5 `decaf::lang::exceptions::InvalidStateException::InvalidStateException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause **Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.422.1.6 `decaf::lang::exceptions::InvalidStateException::InvalidStateException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.422.1.7 `virtual decaf::lang::exceptions::InvalidStateException::~~InvalidStateException () throw () [inline, virtual]`

6.422.2 Member Function Documentation

6.422.2.1 `virtual InvalidStateException* decaf::lang::exceptions::InvalidStateException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/InvalidStateException.h`

6.423 decaf::io::IOException Class Reference

`#include <src/main/decaf/io/IOException.h>`

Inheritance diagram for `decaf::io::IOException`:

Public Member Functions

- **IOException** () throw ()
Default Constructor.
- **IOException** (const **lang::Exception** &ex) throw ()
Copy Constructor.
- **IOException** (const **IOException** &ex) throw ()
Copy Constructor.
- **IOException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IOException** (const std::exception *cause) throw ()
Constructor.
- **IOException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **IOException** * clone () const
Clones this exception.
- virtual ~**IOException** () throw ()

6.423.1 Constructor & Destructor Documentation

6.423.1.1 decaf::io::IOException::IOException () throw () [inline]

Default Constructor.

6.423.1.2 `decaf::io::IOException::IOException (const lang::Exception & ex)
throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy

6.423.1.3 `decaf::io::IOException::IOException (const IOException & ex) throw
() [inline]`

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.423.1.4 `decaf::io::IOException::IOException (const char * file, const int
lineNumber, const std::exception * cause, const char * msg, ...)
throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.423.1.5 `decaf::io::IOException::IOException (const std::exception * cause)
throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.423.1.6 `decaf::io::IOException::IOException (const char * file, const int
lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.423.1.7 `virtual decaf::io::IOException::~~IOException () throw () [inline, virtual]`

6.423.2 Member Function Documentation

6.423.2.1 `virtual IOException* decaf::io::IOException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

- a new instance of an Exception that is a copy of this instance.

Reimplemented from `decaf::lang::Exception` (p. 1715).

Reimplemented in `decaf::internal::net::ssl::openssl::OpenSSLSocketException` (p. 2689), `decaf::io::EOFException` (p. 1709), `decaf::io::InterruptedIOException` (p. 1992), `decaf::io::UnsupportedEncodingException` (p. 3656), `decaf::io::UTFDataFormatException` (p. 3705), `decaf::net::BindException` (p. 770), `decaf::net::ConnectException` (p. 1167), `decaf::net::HttpRetryException` (p. 1861), `decaf::net::MalformedURLException` (p. 2305), `decaf::net::NoRouteToHostException` (p. 2642), `decaf::net::PortUnreachableException` (p. 2783), `decaf::net::ProtocolException` (p. 2939), `decaf::net::SocketException` (p. 3300), `decaf::net::SocketTimeoutException` (p. 3321), `decaf::net::UnknownHostException` (p. 3651), `decaf::net::UnknownServiceException` (p. 3654), and `decaf::util::zip::ZipException` (p. 3798).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/IOException.h`

6.424 activemq::transport::IOTransport Class Reference

Implementation of the **Transport** (p. 3629) interface that performs marshaling of commands to IO streams.

```
#include <src/main/activemq/transport/IOTransport.h>
```

Inheritance diagram for `activemq::transport::IOTransport`:

Public Member Functions

- **IOTransport** ()
Default Constructor.
- **IOTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
*Create an instance of this **Transport** (p. 3629) and assign its WireFormat instance at creation time.*
- virtual ~**IOTransport** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Not supported by this class - throws an exception.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Not supported by this class - throws an exception.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
Sets the WireFormat instance to use.
- virtual void **setTransportListener** (**TransportListener** *listener)
Sets the observer of asynchronous exceptions from this transport.
- virtual **TransportListener** * **getTransportListener** () const
Gets the observer of asynchronous exceptions from this transport.
- virtual void **setInputStream** (decaf::io::DataInputStream *is)
Sets the input stream for in-coming commands.
- virtual void **setOutputStream** (decaf::io::DataOutputStream *os)
Sets the output stream for out-going commands.
- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **stop** () throw (decaf::io::IOException)
*Stops the **Transport** (p. 3629), terminating any threads and stopping all read and write operations.*
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.
- virtual void **run** ()
Runs the polling thread.

- virtual **Transport** * **narrow** (const std::type_info &typeId)

*Narrows down a Chain of Transports to a specific **Transport** (p. 3629) to allow a higher level transport to skip intermediate Transports in certain circumstances.*

- virtual bool **isFaultTolerant** () const

*Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

*Is the **Transport** (p. 3629) Connected to its Broker.*

- virtual bool **isClosed** () const

*Has the **Transport** (p. 3629) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const

- virtual void **reconnect** (const decaf::net::URI &uri AMQCPP_UNUSED) throw (decaf::io::IOException)

reconnect to another location

6.424.1 Detailed Description

Implementation of the **Transport** (p. 3629) interface that performs marshaling of commands to IO streams. This class does not implement the request method, it only handles oneway messages. A thread polls on the input stream for in-coming commands. When a command is received, the command listener is notified. The polling thread is not started until the start method is called. The close method will close the associated streams. Close can be called explicitly by the user, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

6.424.2 Constructor & Destructor Documentation

6.424.2.1 activemq::transport::IOTransport::IOTransport ()

Default Constructor.

6.424.2.2 activemq::transport::IOTransport::IOTransport (const Pointer< wireformat::WireFormat > & wireFormat)

Create an instance of this **Transport** (p. 3629) and assign its WireFormat instance at creation time.

Parameters

wireFormat Data encoder / decoder to use when reading and writing.

6.424.2.3 virtual `activemq::transport::IOTransport::~~IOTransport ()` [virtual]

6.424.3 Member Function Documentation

6.424.3.1 virtual void `activemq::transport::IOTransport::close ()` throw (`decaf::io::IOException`) [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if errors occur.

6.424.3.2 virtual `std::string activemq::transport::IOTransport::getRemoteAddress ()` const [inline, virtual]

Returns

the remote address for this connection

6.424.3.3 virtual `TransportListener* activemq::transport::IOTransport::getTransportListener ()` const [inline, virtual]

Gets the observer of asynchronous exceptions from this transport.

Returns

The listener of transport events.

6.424.3.4 virtual `bool activemq::transport::IOTransport::isClosed ()` const [inline, virtual]

Has the **Transport** (p. 3629) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3629)

6.424.3.5 virtual `bool activemq::transport::IOTransport::isConnected ()` const [inline, virtual]

Is the **Transport** (p. 3629) Connected to its Broker.

Returns

true if a connection has been made.

6.424.3.6 `virtual bool activemq::transport::IOTransport::isFaultTolerant () const`
[inline, virtual]

Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3629) is fault tolerant.

6.424.3.7 `virtual Transport* activemq::transport::IOTransport::narrow (const`
`std::type_info & typeId)` [inline, virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 3629) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

typeId - The type_info of the Object we are searching for.

Returns

the requested Object. or NULL if its not in this chain.

6.424.3.8 `virtual void activemq::transport::IOTransport::oneway (const`
`Pointer< Command > & command) throw (decaf::io::IOException,`
`decaf::lang::exceptions::UnsupportedOperationException)` [virtual]

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

6.424.3.9 `virtual void activemq::transport::IOTransport::reconnect (`
`const decaf::net::URI &uri AMQCPP_UNUSED) throw (`
`decaf::io::IOException)` [inline, virtual]

reconnect to another location

Parameters

uri

Exceptions

IOException on failure of if not supported

6.424.3.10 `virtual Pointer<Response> activemq::transport::IOTransport::request
 (const Pointer< Command > & command
) throw (decaf::io::IOException, de-
 cafe::lang::exceptions::UnsupportedOperationException)
 [virtual]`

Not supported by this class - throws an exception.

Parameters

command the command to be sent.

Returns

the response to the command sent.

Exceptions

UnsupportedOperationException.

6.424.3.11 `virtual Pointer<Response> activemq::transport::IOTransport::request
 (const Pointer< Command > & command, unsigned
 int timeout) throw (decaf::io::IOException,
 decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Not supported by this class - throws an exception.

Parameters

command the command to be sent.

timeout the time to wait for a response.

Returns

the response to the command sent.

Exceptions

UnsupportedOperationException.

6.424.3.12 `virtual void activemq::transport::IOTransport::run () [virtual]`

Runs the polling thread.

Implements `decaf::lang::Runnable` (p. 3112).

6.424.3.13 `virtual void activemq::transport::IOTransport::setInputStream (
 decaf::io::DataInputStream * is) [inline, virtual]`

Sets the input stream for in-coming commands.

Parameters

is The input stream.

6.424.3.14 `virtual void activemq::transport::IOTransport::setOutputStream (decaf::io::DataOutputStream * os) [inline, virtual]`

Sets the output stream for out-going commands.

Parameters

os The output stream.

6.424.3.15 `virtual void activemq::transport::IOTransport::setTransportListener (TransportListener * listener) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

Parameters

listener the listener of transport events.

6.424.3.16 `virtual void activemq::transport::IOTransport::setWireFormat (const Pointer< wireformat::WireFormat > & wireFormat) [inline, virtual]`

Sets the WireFormat instance to use.

Parameters

wireFormat The WireFormat the object used to encode / decode commands.

6.424.3.17 `virtual void activemq::transport::IOTransport::start () throw (decaf::io::IOException) [virtual]`

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions

CMSEException if an error occurs or if this transport has already been closed.

6.424.3.18 `virtual void activemq::transport::IOTransport::stop () throw (decaf::io::IOException) [virtual]`

Stops the **Transport** (p. 3629), terminating any threads and stopping all read and write operations.

Exceptions

IOException if an error occurs while stopping the **Transport** (p. 3629).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/IOTransport.h`

6.425 decaf::lang::Iterable< E > Class Template Reference

Implementing this interface allows an object to be cast to an **Iterable** (p. 2011) type for generic collections API calls.

```
#include <src/main/decaf/lang/Iterable.h>
```

Inheritance diagram for decaf::lang::Iterable< E >:

Public Member Functions

- virtual **~Iterable** ()
- virtual **decaf::util::Iterator< E > * iterator** ()=0
- virtual **decaf::util::Iterator< E > * iterator** () const =0

6.425.1 Detailed Description

```
template<typename E> class decaf::lang::Iterable< E >
```

Implementing this interface allows an object to be cast to an **Iterable** (p. 2011) type for generic collections API calls.

6.425.2 Constructor & Destructor Documentation

6.425.2.1 `template<typename E> virtual decaf::lang::Iterable< E >::~~Iterable () [inline, virtual]`

6.425.3 Member Function Documentation

6.425.3.1 `template<typename E> virtual decaf::util::Iterator<E>* decaf::lang::Iterable< E >::iterator () [pure virtual]`

Returns

an iterator over a set of elements of type T.

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::clear()`, `decaf::util::AbstractCollection< cms::Connection * >::contains()`, `decaf::util::AbstractCollection< cms::Connection * >::copy()`, `decaf::util::AbstractCollection< cms::Connection * >::operator=()`, `decaf::util::AbstractCollection< cms::Connection * >::remove()`, `decaf::util::AbstractSet< ActiveMQSession * >::removeAll()`, `decaf::util::AbstractCollection< cms::Connection * >::removeAll()`, `decaf::util::AbstractCollection< cms::Connection * >::retainAll()`, and `decaf::util::AbstractCollection< cms::Connection * >::toArray()`.

6.425.3.2 `template<typename E> virtual decaf::util::Iterator<E>* decaf::lang::Iterable< E >::iterator () const [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Iterable.h`

6.426 decaf::util::Iterator< T > Class Template Reference

Defines an object that can be used to iterate over the elements of a collection.

```
#include <src/main/decaf/util/Iterator.h>
```

Inheritance diagram for decaf::util::Iterator< T >:

Public Member Functions

- virtual **~Iterator** ()
- virtual T **next** ()=0 throw (lang::exceptions::NoSuchElementException)
Returns the next element in the iteration.
- virtual bool **hasNext** () const =0
Returns true if the iteration has more elements.
- virtual void **remove** ()=0 throw (lang::exceptions::IllegalStateException, lang::exceptions::UnsupportedOperationException)
Removes from the underlying collection the last element returned by the iterator (optional operation).

6.426.1 Detailed Description

```
template<typename T> class decaf::util::Iterator< T >
```

Defines an object that can be used to iterate over the elements of a collection. The iterator provides a way to access and remove elements with well defined semantics.

6.426.2 Constructor & Destructor Documentation

6.426.2.1 `template<typename T> virtual decaf::util::Iterator< T >::~~Iterator ()`
`[inline, virtual]`

6.426.3 Member Function Documentation

6.426.3.1 `template<typename T> virtual bool decaf::util::Iterator< T >::hasNext () const` `[pure virtual]`

Returns true if the iteration has more elements.

Returns false if the next call to next would result in an NoSuchElementException to be thrown.

6.426.3.2 `template<typename T> virtual T decaf::util::Iterator< T >::next ()`
`throw (lang::exceptions::NoSuchElementException)` `[pure virtual]`

Returns the next element in the iteration.

Calling this method repeatedly until the **hasNext()** (p. 2013) method returns false will return each element in the underlying collection exactly once.

Returns

next element in the iteration of elements

Exceptions

NoSuchElementException - iteration has no more elements.

6.426.3.3 `template<typename T> virtual void decaf::util::Iterator< T
>::remove () throw (lang::exceptions::IllegalStateException,
lang::exceptions::UnsupportedOperationException) [pure virtual]`

Removes from the underlying collection the last element returned by the iterator (optional operation).

This method can be called only once per call to next. The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method.

Exceptions

UnsupportedOperationException - if the remove operation is not supported by this **Iterator** (p. 2012).

IllegalStateException - if the next method has not yet been called, or the remove method has already been called after the last call to the next method.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Iterator.h`

6.427 activemq::commands::JournalQueueAck Class Reference

```
#include <src/main/activemq/commands/JournalQueueAck.h>
```

Inheritance diagram for `activemq::commands::JournalQueueAck`:

Public Member Functions

- **JournalQueueAck** ()
- virtual **~JournalQueueAck** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **JournalQueueAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **MessageAck** > & **getMessageAck** () const
- virtual **Pointer**< **MessageAck** > & **getMessageAck** ()
- virtual void **setMessageAck** (const **Pointer**< **MessageAck** > &messageAck)

Static Public Attributes

- static const unsigned char **ID_JOURNALQUEUEACK** = 52

Protected Attributes

- **Pointer**< **ActiveMQDestination** > destination
- **Pointer**< **MessageAck** > messageAck

6.427.1 Constructor & Destructor Documentation

6.427.1.1 **activemq::commands::JournalQueueAck::JournalQueueAck** ()

6.427.1.2 **virtual activemq::commands::JournalQueueAck::~~JournalQueueAck** ()
[virtual]

6.427.2 Member Function Documentation

6.427.2.1 **virtual JournalQueueAck* activemq::commands::JournalQueueAck::cloneDataStructure**
() const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.427.2.2 `virtual void activemq::commands::JournalQueueAck::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1555).

6.427.2.3 `virtual bool activemq::commands::JournalQueueAck::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1556).

6.427.2.4 `virtual unsigned char activemq::commands::JournalQueueAck::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.427.2.5 `virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination () const` [virtual]
- 6.427.2.6 `virtual Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination ()` [virtual]
- 6.427.2.7 `virtual const Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck () const` [virtual]
- 6.427.2.8 `virtual Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck ()` [virtual]
- 6.427.2.9 `virtual void activemq::commands::JournalQueueAck::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.427.2.10 `virtual void activemq::commands::JournalQueueAck::setMessageAck (const Pointer< MessageAck > & messageAck)` [virtual]
- 6.427.2.11 `virtual std::string activemq::commands::JournalQueueAck::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.767).

6.427.3 Field Documentation

- 6.427.3.1 `Pointer<ActiveMQDestination> activemq::commands::JournalQueueAck::destination` [protected]
- 6.427.3.2 `const unsigned char activemq::commands::JournalQueueAck::ID_-JOURNALQUEUEACK = 52` [static]
- 6.427.3.3 `Pointer<MessageAck> activemq::commands::JournalQueueAck::messageAck` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalQueueAck.h`

6.428 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2017).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.428.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2017). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.428.2 Constructor & Destructor Documentation

6.428.2.1 `activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::JournalQueueAckMarshaller () [inline]`

6.428.2.2 `virtual activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller () [inline, virtual]`

6.428.3 Member Function Documentation

6.428.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.428.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.428.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.428.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.428.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.428.3.6 virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.428.3.7 virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAckMarshaller.h`

6.429 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for `JournalQueueAckMarshaller` (p. 2021).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.429.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2021). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.429.2 Constructor & Destructor Documentation

6.429.2.1 `activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::JournalQueueAckMarshaller () [inline]`

6.429.2.2 `virtual activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller () [inline, virtual]`

6.429.3 Member Function Documentation

6.429.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.429.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.429.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.429.3.4 virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.429.3.5 virtual int activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.429.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.429.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h`

6.430 **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2025).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.430.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2025). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.430.2 Constructor & Destructor Documentation

6.430.2.1 `activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::JournalQueueAckMarshaller () [inline]`

6.430.2.2 `virtual activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller () [inline, virtual]`

6.430.3 Member Function Documentation

6.430.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.430.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.430.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.430.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.430.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.430.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.430.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h`

6.431 **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2028).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.431.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.2028). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.431.2 Constructor & Destructor Documentation

6.431.2.1 `activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::JournalQueueAckMarshaller () [inline]`

6.431.2.2 `virtual activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller () [inline, virtual]`

6.431.3 Member Function Documentation

6.431.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.431.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.431.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.431.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.431.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.431.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.431.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h`

6.432 **activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2032).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.432.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2032). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.432.2 Constructor & Destructor Documentation

6.432.2.1 `activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::JournalQueueAckMarshaller () [inline]`

6.432.2.2 `virtual activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller () [inline, virtual]`

6.432.3 Member Function Documentation

6.432.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.432.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.432.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.432.3.4 virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.432.3.5 virtual int activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.432.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.432.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h`

6.433 **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2036).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.433.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2036). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.433.2 Constructor & Destructor Documentation

6.433.2.1 `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::JournalQueueAckMarshaller () [inline]`

6.433.2.2 `virtual activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller () [inline, virtual]`

6.433.3 Member Function Documentation

6.433.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.433.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.433.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.433.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.433.3.5 virtual int activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.433.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

```
6.433.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h`

6.434 `activemq::commands::JournalTopicAck` Class Reference

```
#include <src/main/activemq/commands/JournalTopicAck.h>
```

Inheritance diagram for `activemq::commands::JournalTopicAck`:

Public Member Functions

- `JournalTopicAck ()`
- `virtual ~JournalTopicAck ()`

- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **JournalTopicAck** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual long long **getMessageSequenceId** () const
- virtual void **setMessageSequenceId** (long long messageSequenceId)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)

Static Public Attributes

- static const unsigned char **ID_JOURNALTOPICACK** = 50

Protected Attributes

- **Pointer**< **ActiveMQDestination** > destination
- **Pointer**< **MessageId** > messageId
- long long messageSequenceId
- std::string subscriptionName
- std::string clientId
- **Pointer**< **TransactionId** > transactionId

6.434.1 Constructor & Destructor Documentation

6.434.1.1 `activemq::commands::JournalTopicAck::JournalTopicAck ()`

6.434.1.2 `virtual activemq::commands::JournalTopicAck::~~JournalTopicAck ()`
[virtual]

6.434.2 Member Function Documentation

6.434.2.1 `virtual JournalTopicAck* activemq::commands::JournalTopicAck::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.434.2.2 `virtual void activemq::commands::JournalTopicAck::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1555).

6.434.2.3 `virtual bool activemq::commands::JournalTopicAck::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1556).

6.434.2.4 `virtual const std::string& activemq::commands::JournalTopicAck::getClientId () const [virtual]`

6.434.2.5 `virtual std::string& activemq::commands::JournalTopicAck::getClientId () [virtual]`

6.434.2.6 `virtual unsigned char activemq::commands::JournalTopicAck::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSetructure** (p. 1553) type copy.

Implements **activemq::commands::DataSetructure** (p. 1557).

-
- 6.434.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination () const` [virtual]
 - 6.434.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination ()` [virtual]
 - 6.434.2.9 `virtual const Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId () const` [virtual]
 - 6.434.2.10 `virtual Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId ()` [virtual]
 - 6.434.2.11 `virtual long long activemq::commands::JournalTopicAck::getMessageSequenceId () const` [virtual]
 - 6.434.2.12 `virtual std::string& activemq::commands::JournalTopicAck::getSubscriptionName ()` [virtual]
 - 6.434.2.13 `virtual const std::string& activemq::commands::JournalTopicAck::getSubscriptionName () const` [virtual]
 - 6.434.2.14 `virtual const Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId () const` [virtual]
 - 6.434.2.15 `virtual Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId ()` [virtual]
 - 6.434.2.16 `virtual void activemq::commands::JournalTopicAck::setClientId (const std::string & clientId)` [virtual]
 - 6.434.2.17 `virtual void activemq::commands::JournalTopicAck::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
 - 6.434.2.18 `virtual void activemq::commands::JournalTopicAck::setMessageId (const Pointer< MessageId > & messageId)` [virtual]
 - 6.434.2.19 `virtual void activemq::commands::JournalTopicAck::setMessageSequenceId (long long messageSequenceId)` [virtual]
 - 6.434.2.20 `virtual void activemq::commands::JournalTopicAck::setSubscriptionName (const std::string & subscriptionName)` [virtual]
 - 6.434.2.21 `virtual void activemq::commands::JournalTopicAck::setTransactionId (const Pointer< TransactionId > & transactionId)` [virtual]
 - 6.434.2.22 `virtual std::string activemq::commands::JournalTopicAck::toString () const` [virtual]
-

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 767).

6.434.3 Field Documentation

- 6.434.3.1 `std::string activemq::commands::JournalTopicAck::clientId` [protected]
- 6.434.3.2 `Pointer<ActiveMQDestination> activemq::commands::JournalTopicAck::destination` [protected]
- 6.434.3.3 `const unsigned char activemq::commands::JournalTopicAck::ID_ - JOURNALTOPICACK = 50` [static]
- 6.434.3.4 `Pointer<MessageId> activemq::commands::JournalTopicAck::messageId` [protected]
- 6.434.3.5 `long long activemq::commands::JournalTopicAck::messageSequenceId` [protected]
- 6.434.3.6 `std::string activemq::commands::JournalTopicAck::subscriptionName` [protected]
- 6.434.3.7 `Pointer<TransactionId> activemq::commands::JournalTopicAck::transactionId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTopicAck.h`

6.435 `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMa` Class Reference

Marshaling code for Open Wire Format for `JournalTopicAckMarshaller` (p. 2045).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller`:

Public Member Functions

- `JournalTopicAckMarshaller ()`
- `virtual ~JournalTopicAckMarshaller ()`
- `virtual commands::DataStructure * createObject () const`

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.435.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.2045). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.435.2 Constructor & Destructor Documentation

6.435.2.1 `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::JournalTopicAckMarshaller () [inline]`

6.435.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller () [inline, virtual]`

6.435.3 Member Function Documentation

6.435.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.435.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.435.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.435.3.4 virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.435.3.5 virtual int activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.435.3.6 virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.435.3.7 virtual void ac-
 tivemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h

6.436 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2049).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.436.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.2049). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.436.2 Constructor & Destructor Documentation

6.436.2.1 `activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::JournalTopicAckMarshaller () [inline]`

6.436.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller () [inline, virtual]`

6.436.3 Member Function Documentation

6.436.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.436.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.436.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.436.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.436.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.436.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.436.3.7 virtual void ac-
 tivemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h

6.437 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2053).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.437.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.2053). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.437.2 Constructor & Destructor Documentation

6.437.2.1 `activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

6.437.2.2 `virtual activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

6.437.3 Member Function Documentation

6.437.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.437.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.437.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.437.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.437.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.437.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.437.3.7 virtual void ac-
 tivemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h

6.438 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2057).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.438.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.2057). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.438.2 Constructor & Destructor Documentation

6.438.2.1 `activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

6.438.2.2 `virtual activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

6.438.3 Member Function Documentation

6.438.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.438.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.438.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.438.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.438.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.438.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.438.3.7 virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h

6.439 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2061).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.439.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.2061). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.439.2 Constructor & Destructor Documentation

6.439.2.1 `activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

6.439.2.2 `virtual activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

6.439.3 Member Function Documentation

6.439.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.439.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.439.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.439.3.4 virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.439.3.5 virtual int activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.439.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.439.3.7 virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h

6.440 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2065).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.440.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.2065). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.440.2 Constructor & Destructor Documentation

6.440.2.1 `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

6.440.2.2 `virtual activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

6.440.3 Member Function Documentation

6.440.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.440.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.440.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.440.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.440.3.5 virtual int activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.440.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.440.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTopicAckMarshaller.h**

6.441 activemq::commands::JournalTrace Class Reference

```
#include <src/main/activemq/commands/JournalTrace.h>
```

Inheritance diagram for **activemq::commands::JournalTrace**:

Public Member Functions

- **JournalTrace ()**
- virtual **~JournalTrace ()**
- virtual unsigned char **getDataStructureType ()** const

Get the unique identifier that this object and its own Marshaler share.

- virtual **JournalTrace** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getMessage** () const
- virtual std::string & **getMessage** ()
- virtual void **setMessage** (const std::string &message)

Static Public Attributes

- static const unsigned char **ID_JOURNALTRACE** = 53

Protected Attributes

- std::string **message**

6.441.1 Constructor & Destructor Documentation

6.441.1.1 **activemq::commands::JournalTrace::JournalTrace** ()

6.441.1.2 **virtual activemq::commands::JournalTrace::~~JournalTrace** ()
 [virtual]

6.441.2 Member Function Documentation

6.441.2.1 **virtual JournalTrace* activemq::commands::JournalTrace::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.441.2.2 virtual void activemq::commands::JournalTrace::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1555).

6.441.2.3 virtual bool activemq::commands::JournalTrace::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1556).

6.441.2.4 virtual unsigned char activemq::commands::JournalTrace::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

6.441.2.5 virtual std::string& activemq::commands::JournalTrace::getMessage () [virtual]**6.441.2.6 virtual const std::string& activemq::commands::JournalTrace::getMessage () const [virtual]****6.441.2.7 virtual void activemq::commands::JournalTrace::setMessage (const std::string & *message*) [virtual]****6.441.2.8 virtual std::string activemq::commands::JournalTrace::toString () const [virtual]**

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 767).

6.441.3 Field Documentation

6.441.3.1 `const unsigned char activemq::commands::JournalTrace::ID_-
JOURNALTRACE = 53` [static]

6.441.3.2 `std::string activemq::commands::JournalTrace::message` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTrace.h`

6.442 `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` Class Reference

Marshaling code for Open Wire Format for `JournalTraceMarshaller` (p. 2072).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller`:

Public Member Functions

- `JournalTraceMarshaller ()`
- `virtual ~JournalTraceMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.442.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.2072). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.442.2 Constructor & Destructor Documentation

6.442.2.1 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::JournalTraceMarshaller () [inline]

6.442.2.2 virtual activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::~~JournalTraceMarshaller () [inline, virtual]

6.442.3 Member Function Documentation

6.442.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.442.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.442.3.3 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.442.3.4 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.442.3.5 virtual int activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.442.3.6 virtual void **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal2**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.442.3.7 virtual void **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h`

6.443 **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2076).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller**:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.443.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.2076). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.443.2 Constructor & Destructor Documentation

6.443.2.1 **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::JournalTraceMarshaller**
() [inline]

6.443.2.2 **virtual**
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::~~JournalTraceMarshaller
() [inline, virtual]

6.443.3 Member Function Documentation

6.443.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.443.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::getDataStructureType**
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.443.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseMarshal**
(**OpenWireFormat * wireFormat**, **commands::DataStructure * dataStructure**, **decaf::io::DataOutputStream * dataOut**) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

```
6.443.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

```
6.443.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.443.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.443.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h`

6.444 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller Class Reference

Marshaling code for Open Wire Format for `JournalTraceMarshaller` (p. 2079).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller`:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.444.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.2079). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.444.2 Constructor & Destructor Documentation

6.444.2.1 `activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

6.444.2.2 `virtual activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

6.444.3 Member Function Documentation

6.444.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.444.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.444.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::looseMarshal(OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.444.3.4 virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.444.3.5 virtual int activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.444.3.6 virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.444.3.7 virtual void **activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h`

6.445 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2083).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller**:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.445.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.2083). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.445.2 Constructor & Destructor Documentation

6.445.2.1 `activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

6.445.2.2 `virtual activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

6.445.3 Member Function Documentation

6.445.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.445.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.445.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.445.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.445.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.445.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.445.3.7 virtual void **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h`

6.446 **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2087).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller**:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.446.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.2087). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.446.2 Constructor & Destructor Documentation

6.446.2.1 `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::JournalTraceMarshaller () [inline]`

6.446.2.2 `virtual activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::~~JournalTraceMarshaller () [inline, virtual]`

6.446.3 Member Function Documentation

6.446.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.446.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.446.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.446.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.446.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.446.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.446.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTraceMarshaller.h**

6.447 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2091).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.447.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.2091). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.447.2 Constructor & Destructor Documentation

6.447.2.1 `activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::JournalTraceMarshaller () [inline]`

6.447.2.2 `virtual activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::~~JournalTraceMarshaller () [inline, virtual]`

6.447.3 Member Function Documentation

6.447.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.447.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.447.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.447.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.447.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.447.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

```
6.447.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h`

6.448 activemq::commands::JournalTransaction Class Reference

```
#include <src/main/activemq/commands/JournalTransaction.h>
```

Inheritance diagram for `activemq::commands::JournalTransaction`:

Public Member Functions

- `JournalTransaction ()`
- `virtual ~JournalTransaction ()`

- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **JournalTransaction** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **getWasPrepared** () const
- virtual void **setWasPrepared** (bool wasPrepared)

Static Public Attributes

- static const unsigned char **ID_JOURNALTRANSACTION** = 54

Protected Attributes

- **Pointer**< **TransactionId** > transactionId
- unsigned char type
- bool wasPrepared

6.448.1 Constructor & Destructor Documentation

6.448.1.1 **activemq::commands::JournalTransaction::JournalTransaction** ()

6.448.1.2 **virtual activemq::commands::JournalTransaction::~~JournalTransaction** () [virtual]

6.448.2 Member Function Documentation

6.448.2.1 **virtual JournalTransaction* activemq::commands::JournalTransaction::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.448.2.2 `virtual void activemq::commands::JournalTransaction::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1555).

6.448.2.3 `virtual bool activemq::commands::JournalTransaction::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1556).

6.448.2.4 `virtual unsigned char activemq::commands::JournalTransaction::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.448.2.5 `virtual const Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId () const [virtual]`
- 6.448.2.6 `virtual Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId () [virtual]`
- 6.448.2.7 `virtual unsigned char activemq::commands::JournalTransaction::getType () const [virtual]`
- 6.448.2.8 `virtual bool activemq::commands::JournalTransaction::getWasPrepared () const [virtual]`
- 6.448.2.9 `virtual void activemq::commands::JournalTransaction::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`
- 6.448.2.10 `virtual void activemq::commands::JournalTransaction::setType (unsigned char type) [virtual]`
- 6.448.2.11 `virtual void activemq::commands::JournalTransaction::setWasPrepared (bool wasPrepared) [virtual]`
- 6.448.2.12 `virtual std::string activemq::commands::JournalTransaction::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.767).

6.448.3 Field Documentation

- 6.448.3.1 `const unsigned char activemq::commands::JournalTransaction::ID_ - JOURNALTRANSACTION = 54 [static]`
- 6.448.3.2 `Pointer<TransactionId> activemq::commands::JournalTransaction::transactionId [protected]`
- 6.448.3.3 `unsigned char activemq::commands::JournalTransaction::type [protected]`
- 6.448.3.4 `bool activemq::commands::JournalTransaction::wasPrepared [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTransaction.h`

6.449 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2099).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTransactionMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.449.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.2099). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.449.2 Constructor & Destructor Documentation

6.449.2.1 `activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::JournalTrans`
() [inline]

6.449.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::~~JournalTran`
() [inline, virtual]

6.449.3 Member Function Documentation

6.449.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::createObject`
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1505).

6.449.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::getDataStructur`
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1511).

6.449.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::looseMarshal`
(`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.449.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.449.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.449.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.449.3.7 `virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/JournalTransactionMarshaller.h`

6.450 `activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` Class Reference

Marshaling code for Open Wire Format for `JournalTransactionMarshaller` (p. 2102).

#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.450.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.2102). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.450.2 Constructor & Destructor Documentation

6.450.2.1 `activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::JournalTransactionMarshaller () [inline]`

6.450.2.2 `virtual activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::~~JournalTransactionMarshaller () [inline, virtual]`

6.450.3 Member Function Documentation

6.450.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.450.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.450.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.450.3.4 virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.450.3.5 virtual int activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.450.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.450.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h`

6.451 **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2106).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.451.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.2106). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.451.2 Constructor & Destructor Documentation

6.451.2.1 `activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::JournalTrans`
`() [inline]`

6.451.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::~~JournalTran`
`() [inline, virtual]`

6.451.3 Member Function Documentation

6.451.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.451.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.451.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.451.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.451.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.451.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.451.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h`

6.452 **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2110).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.452.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.2110). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.452.2 Constructor & Destructor Documentation

6.452.2.1 `activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::JournalTrans`
`() [inline]`

6.452.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::~~JournalTran`
`() [inline, virtual]`

6.452.3 Member Function Documentation

6.452.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.452.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.452.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.452.3.4 virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.452.3.5 virtual int activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.452.3.6 virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.452.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h`

6.453 **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2114).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.453.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.2114). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.453.2 Constructor & Destructor Documentation

6.453.2.1 `activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::JournalTrans`
`() [inline]`

6.453.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::~~JournalTran`
`() [inline, virtual]`

6.453.3 Member Function Documentation

6.453.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.453.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.453.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.453.3.4 virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.453.3.5 virtual int activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.453.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.453.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h`

6.454 **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2118).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.454.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.2118). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.454.2 Constructor & Destructor Documentation

6.454.2.1 `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::JournalTrans`
`() [inline]`

6.454.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::~~JournalTran`
`() [inline, virtual]`

6.454.3 Member Function Documentation

6.454.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.454.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.454.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.454.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.454.3.5 virtual int activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.454.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.454.3.7 virtual void **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h`

6.455 activemq::commands::KeepAliveInfo Class Reference

```
#include <src/main/activemq/commands/KeepAliveInfo.h>
```

Inheritance diagram for **activemq::commands::KeepAliveInfo**:

Public Member Functions

- **KeepAliveInfo** ()
- virtual **~KeepAliveInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **KeepAliveInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual bool **isKeepAliveInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_KEEPAALIVEINFO** = 10

6.455.1 Constructor & Destructor Documentation

6.455.1.1 **activemq::commands::KeepAliveInfo::KeepAliveInfo** ()

6.455.1.2 **virtual activemq::commands::KeepAliveInfo::~~KeepAliveInfo** ()
 [virtual]

6.455.2 Member Function Documentation

6.455.2.1 **virtual KeepAliveInfo* activemq::commands::KeepAliveInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.455.2.2 `virtual void activemq::commands::KeepAliveInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.455.2.3 `virtual bool activemq::commands::KeepAliveInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.455.2.4 `virtual unsigned char activemq::commands::KeepAliveInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1553) type copy.

Implements `activemq::commands::DataStructure` (p. 1557).

6.455.2.5 `virtual bool activemq::commands::KeepAliveInfo::isKeepAliveInfo () const [inline, virtual]`

Returns

an answer of true to the `isKeepAliveInfo()` (p. 2124) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 698).

6.455.2.6 `virtual std::string activemq::commands::KeepAliveInfo::toString () const [virtual]`

Returns a string containing the information for this `DataStructure` (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 700).

6.455.2.7 `virtual Pointer<Command> activemq::commands::KeepAliveInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.455.3 Field Documentation

6.455.3.1 `const unsigned char activemq::commands::KeepAliveInfo::ID_ - KEEPALIVEINFO = 10` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/KeepAliveInfo.h`

6.456 **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2125).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller**:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.456.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2125). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.456.2 Constructor & Destructor Documentation

6.456.2.1 **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::KeepAliveInfoMar**
() [inline]

6.456.2.2 virtual
activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]

6.456.3 Member Function Documentation

6.456.3.1 virtual **commands::DataStructure*** **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.456.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.456.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.456.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

6.456.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

6.456.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

6.456.3.7 `virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h`

6.457 **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2129).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller**:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.457.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2129). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.457.2 Constructor & Destructor Documentation

6.457.2.1 **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::KeepAliveInfoMar**
() [inline]

6.457.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]

6.457.3 Member Function Documentation

6.457.3.1 **virtual commands::DataStructure* ac-**
tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.457.3.2 **virtual unsigned char ac-**
tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.457.3.3 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 736).

6.457.3.4 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 737).

6.457.3.5 virtual int activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

```
6.457.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

```
6.457.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h`

6.458 `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `KeepAliveInfoMarshaller` (p. 2133).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller`:

Public Member Functions

- `KeepAliveInfoMarshaller ()`
- `virtual ~KeepAliveInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.458.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2133). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.458.2 Constructor & Destructor Documentation

6.458.2.1 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]

6.458.2.2 virtual
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]

6.458.3 Member Function Documentation

6.458.3.1 virtual commands::DataStructure* ac-
tivismq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.458.3.2 virtual unsigned char ac-
tivismq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.458.3.3 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.458.3.4 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.458.3.5 virtual int activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

```
6.458.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.458.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h`

6.459 `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `KeepAliveInfoMarshaller` (p. 2137).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller`:

Public Member Functions

- `KeepAliveInfoMarshaller ()`
- `virtual ~KeepAliveInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.459.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2137). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.459.2 Constructor & Destructor Documentation

6.459.2.1 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]

6.459.2.2 virtual
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]

6.459.3 Member Function Documentation

6.459.3.1 virtual commands::DataStructure* ac-
tivismq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.459.3.2 virtual unsigned char ac-
tivismq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.459.3.3 virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 709).

6.459.3.4 virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 710).

6.459.3.5 virtual int activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

```
6.459.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.459.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h`

6.460 `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `KeepAliveInfoMarshaller` (p. 2141).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller`:

Public Member Functions

- `KeepAliveInfoMarshaller ()`
- `virtual ~KeepAliveInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.460.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2141). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.460.2 Constructor & Destructor Documentation

6.460.2.1 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]

6.460.2.2 virtual
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]

6.460.3 Member Function Documentation

6.460.3.1 virtual commands::DataStructure* ac-
tivismq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.460.3.2 virtual unsigned char ac-
tivismq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.460.3.3 virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.460.3.4 virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

6.460.3.5 virtual int activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```
6.460.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.460.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h`

6.461 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2145).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller**:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.461.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2145). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.461.2 Constructor & Destructor Documentation

6.461.2.1 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]

6.461.2.2 virtual
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]

6.461.3 Member Function Documentation

6.461.3.1 virtual commands::DataStructure* ac-
tivismq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.461.3.2 virtual unsigned char ac-
tivismq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.461.3.3 virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.461.3.4 virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.461.3.5 virtual int activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

```
6.461.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

```
6.461.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h`

6.462 decaf::security::Key Class Reference

The **Key** (p. 2149) interface is the top-level interface for all keys.

```
#include <src/main/decaf/security/Key.h>
```

Inheritance diagram for `decaf::security::Key`:

Public Member Functions

- virtual **~Key** ()
- virtual `std::string` **getAlgorithm** () const =0
Returns the standard algorithm name for this key.
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0
Provides the key in its primary encoding format, or nothing if this key does not support encoding.
- virtual `std::string` **getFormat** () const =0
Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.

6.462.1 Detailed Description

The **Key** (p. 2149) interface is the top-level interface for all keys. It defines the functionality shared by all key objects. All keys have three characteristics:

An Algorithm

This is the key algorithm for that key. The key algorithm is usually an encryption or asymmetric operation algorithm (such as DSA or RSA), which will work with those algorithms and with related algorithms (such as MD5 with RSA, SHA-1 with RSA, Raw DSA, etc.) The name of the algorithm of a key is obtained using the `getAlgorithm` method.

An Encoded Form

This is an external encoded form for the key used when a standard representation of the key is needed outside the application, as when transmitting the key to some other party. The key is encoded according to a standard format (such as X.509 `SubjectPublicKeyInfo` or PKCS#8), and is returned using the `getEncoded` method. Note: The syntax of the ASN.1 type `SubjectPublicKeyInfo` is defined as follows:

SubjectPublicKeyInfo ::= SEQUENCE {

algorithm AlgorithmIdentifier,

subjectPublicKey BIT STRING }

AlgorithmIdentifier ::= SEQUENCE {

algorithm OBJECT IDENTIFIER,

parameters ANY DEFINED BY algorithm OPTIONAL }

For more information, see RFC 2459: Internet X.509 Public **Key** (p.2149) Infrastructure Certificate and CRL Profile.

A Format

This is the name of the format of the encoded key. It is returned by the `getFormat` method.

6.462.2 Constructor & Destructor Documentation

6.462.2.1 `virtual decaf::security::Key::~~Key () [inline, virtual]`

6.462.3 Member Function Documentation

6.462.3.1 `virtual std::string decaf::security::Key::getAlgorithm () const [pure virtual]`

Returns the standard algorithm name for this key.

For example, "DSA" would indicate that this key is a DSA key.

Returns

the name of the algorithm associated with this key.

6.462.3.2 `virtual void decaf::security::Key::getEncoded (std::vector< unsigned char > & output) const [pure virtual]`

Provides the key in its primary encoding format, or nothing if this key does not support encoding.

Parameters

output Receives the encoded key, or nothing if the key does not support encoding.

6.462.3.3 `virtual std::string decaf::security::Key::getFormat () const [pure virtual]`

Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.

The primary encoding format is named in terms of the appropriate ASN.1 data format, if an ASN.1 specification for this key exists. For example, the name of the ASN.1 data format for public keys is SubjectPublicKeyInfo, as defined by the X.509 standard; in this case, the returned format is "X.509". Similarly, the name of the ASN.1 data format for private keys is PrivateKeyInfo, as defined by the PKCS #8 standard; in this case, the returned format is "PKCS#8".

Returns

the primary encoding format of the key.

The documentation for this class was generated from the following file:

- src/main/decaf/security/**Key.h**

6.463 decaf::security::KeyException Class Reference

```
#include <src/main/decaf/security/KeyException.h>
```

Inheritance diagram for decaf::security::KeyException:

Public Member Functions

- **KeyException** () throw ()
Default Constructor.
- **KeyException** (const decaf::lang::Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **KeyException** (const **KeyException** &ex) throw ()
Copy Constructor.
- **KeyException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **KeyException** (const std::exception *cause) throw ()
Constructor.
- **KeyException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **KeyException** * clone () const
Clones this exception.
- virtual ~**KeyException** () throw ()

6.463.1 Constructor & Destructor Documentation**6.463.1.1 decaf::security::KeyException::KeyException () throw () [inline]**

Default Constructor.

6.463.1.2 `decaf::security::KeyException::KeyException (const
decaf::lang::Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.463.1.3 `decaf::security::KeyException::KeyException (const KeyException &
ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.463.1.4 `decaf::security::KeyException::KeyException (const char * file, const
int lineNumber, const std::exception * cause, const char * msg, ...
) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.463.1.5 `decaf::security::KeyException::KeyException (const std::exception *
cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.463.1.6 `decaf::security::KeyException::KeyException (const char * file, const
int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

- file* name where exception occurs
- lineNumber* line number where the exception occurred.
- msg* message to report
- ... list of primitives that are formatted into the message

6.463.1.7 virtual decaf::security::KeyException::~~KeyException () throw ()
[inline, virtual]

6.463.2 Member Function Documentation

6.463.2.1 virtual KeyException* decaf::security::KeyException::clone () const
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from decaf::security::GeneralSecurityException (p. 1847).

Reimplemented in decaf::security::InvalidKeyException (p. 1996), and decaf::security::KeyManagementException (p. 2155).

The documentation for this class was generated from the following file:

- src/main/decaf/security/KeyException.h

6.464 decaf::security::KeyManagementException Class Reference

```
#include <src/main/decaf/security/KeyManagementException.h>
```

Inheritance diagram for decaf::security::KeyManagementException:

Public Member Functions

- **KeyManagementException** () throw ()
Default Constructor.
- **KeyManagementException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **KeyManagementException** (const KeyManagementException &ex) throw ()

Copy Constructor.

- **KeyManagementException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- **KeyManagementException** (const std::exception *cause) throw ()

Constructor.

- **KeyManagementException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **KeyManagementException** * clone () const

Clones this exception.

- virtual ~**KeyManagementException** () throw ()

6.464.1 Constructor & Destructor Documentation

6.464.1.1 decaf::security::KeyManagementException::KeyManagementException () throw () [inline]

Default Constructor.

6.464.1.2 decaf::security::KeyManagementException::KeyManagementException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.464.1.3 decaf::security::KeyManagementException::KeyManagementException (const KeyManagementException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.464.1.4 decaf::security::KeyManagementException::KeyManagementException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.464.1.5 `decaf::security::KeyManagementException::KeyManagementException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.464.1.6 `decaf::security::KeyManagementException::KeyManagementException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs
lineNumber line number where the exception occurred.
msg message to report
... list of primitives that are formatted into the message

6.464.1.7 `virtual decaf::security::KeyManagementException::~KeyManagementException () throw () [inline, virtual]`

6.464.2 Member Function Documentation

6.464.2.1 `virtual KeyManagementException* decaf::security::KeyManagementException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::KeyException** (p. 2153).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/KeyManagementException.h`

6.465 **activemq::commands::LastPartialCommand** Class Reference

```
#include <src/main/activemq/commands/LastPartialCommand.h>
```

Inheritance diagram for **activemq::commands::LastPartialCommand**:

Public Member Functions

- **LastPartialCommand** ()
- virtual **~LastPartialCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **LastPartialCommand * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

Static Public Attributes

- static const unsigned char **ID_LASTPARTIALCOMMAND** = 61

6.465.1 Constructor & Destructor Documentation

6.465.1.1 `activemq::commands::LastPartialCommand::LastPartialCommand ()`

6.465.1.2 `virtual
activemq::commands::LastPartialCommand::~~LastPartialCommand ()
[virtual]`

6.465.2 Member Function Documentation

6.465.2.1 `virtual LastPartialCommand* ac-
tivemq::commands::LastPartialCommand::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::PartialCommand` (p. 2729).

6.465.2.2 `virtual void ac-
tivemq::commands::LastPartialCommand::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::PartialCommand` (p. 2729).

6.465.2.3 `virtual bool activemq::commands::LastPartialCommand::equals (const
DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::PartialCommand` (p. 2729).

6.465.2.4 `virtual unsigned char ac-
tivemq::commands::LastPartialCommand::getDataStructureType ()
const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1553) type copy.

Reimplemented from **activemq::commands::PartialCommand** (p. 2730).

6.465.2.5 `virtual std::string activemq::commands::LastPartialCommand::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::PartialCommand** (p. 2730).

6.465.3 Field Documentation

6.465.3.1 `const unsigned char activemq::commands::LastPartialCommand::ID_ - LASTPARTIALCOMMAND = 61 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/LastPartialCommand.h`

6.466 **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2158).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/LastPartialCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller**:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataSet * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataSetType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.466.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.2158).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.466.2 Constructor & Destructor Documentation

- 6.466.2.1 **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::LastPartialCommandMarshaller** () [inline]
- 6.466.2.2 **virtual activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::~LastPartialCommandMarshaller** () [inline, virtual]

6.466.3 Member Function Documentation

- 6.466.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2732).

6.466.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaller.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2732).

6.466.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2733).

6.466.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2733).

6.466.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2734).

6.466.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2734).

```

6.466.3.7 virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 2735).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/LastPartialCommandMarshaller.h`

6.467 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2162).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.467.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.2162).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.467.2 Constructor & Destructor Documentation

6.467.2.1 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::LastPartialCommandMarshaller () [inline]

6.467.2.2 virtual
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller () [inline, virtual]

6.467.3 Member Function Documentation

6.467.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2745).

6.467.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::getDataStructureType() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2745).

6.467.3.3 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseMarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2745).

6.467.3.4 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseUnmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2746).

6.467.3.5 virtual int activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2746).

6.467.3.6 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2747).

6.467.3.7 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2747).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h`

6.468 `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `LastPartialCommandMarshaller` (p. 2166).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`:

Public Member Functions

- `LastPartialCommandMarshaller ()`
- `virtual ~LastPartialCommandMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.468.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.2166).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.468.2 Constructor & Destructor Documentation

6.468.2.1 **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::LastPartialCommandMarshaller** () [inline]

6.468.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::~LastPartialCommandMarshaller () [inline, virtual]

6.468.3 Member Function Documentation

6.468.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2736).

6.468.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2737).

6.468.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2737).

6.468.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2737).

6.468.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2738).

6.468.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2738).

6.468.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2739).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h`

6.469 `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `LastPartialCommandMarshaller` (p. 2170).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`:

Public Member Functions

- `LastPartialCommandMarshaller ()`
- `virtual ~LastPartialCommandMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.469.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.2170).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.469.2 Constructor & Destructor Documentation

6.469.2.1 `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.469.2.2 `virtual activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

6.469.3 Member Function Documentation

6.469.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2741).

6.469.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2741).

6.469.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2741).

6.469.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2742).

6.469.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2742).

6.469.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2743).

6.469.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2743).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h`

6.470 `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `LastPartialCommandMarshaller` (p. 2174).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`:

Public Member Functions

- `LastPartialCommandMarshaller ()`
- `virtual ~LastPartialCommandMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.470.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.2174).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.470.2 Constructor & Destructor Documentation

6.470.2.1 `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.470.2.2 `virtual activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

6.470.3 Member Function Documentation

6.470.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2749).

6.470.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2749).

6.470.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2750).

6.470.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2750).

6.470.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2750).

6.470.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2751).

6.470.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2751).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h`

6.471 `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `LastPartialCommandMarshaller` (p. 2178).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller`:

Public Member Functions

- `LastPartialCommandMarshaller ()`
- `virtual ~LastPartialCommandMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.471.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.2178).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.471.2 Constructor & Destructor Documentation

6.471.2.1 `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.471.2.2 `virtual activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

6.471.3 Member Function Documentation

6.471.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2753).

6.471.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2753).

6.471.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2754).

6.471.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2754).

6.471.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2755).

6.471.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2755).

6.471.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**LastPartialCommandMarshaller.h**

6.472 decaf::util::comparators::Less< E > Class Template Reference

Simple **Less** (p. 2182) **Comparator** (p. 1127) that compares to elements to determine if the first is less than the second.

```
#include <src/main/decaf/util/comparators/Less.h>
```

Inheritance diagram for decaf::util::comparators::Less< E >:

Public Member Functions

- **Less** ()
- virtual **~Less** ()
- virtual bool **operator**() (const E &left, const E &right) const
*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1127) to be passed to an STL **Map** (p. 2305) for use as the sorting criteria.*
- virtual int **compare** (const E &o1, const E &o2) const
Compares its two arguments for order.

6.472.1 Detailed Description

```
template<typename E> class decaf::util::comparators::Less< E >
```

Simple **Less** (p. 2182) **Comparator** (p. 1127) that compares to elements to determine if the first is less than the second. This can be used in **Collection** (p. 1097) classes to sort elements according to their natural ordering. By design the Comparator's compare function return more information about comparison than the STL binary function's boolean compare operator. In this case the compare method will return

Since

1.0

6.472.2 Constructor & Destructor Documentation

6.472.2.1 `template<typename E > decaf::util::comparators::Less< E >::Less ()`
[inline]

6.472.2.2 `template<typename E > virtual decaf::util::comparators::Less< E >::~~Less ()` [inline, virtual]

6.472.3 Member Function Documentation

6.472.3.1 `template<typename E > virtual int decaf::util::comparators::Less< E >::compare (const E & o1, const E & o2) const` [inline, virtual]

Compares its two arguments for order.

Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that `sgn(compare(x, y)) == -sgn(compare(y, x))` for all x and y. (This implies that `compare(x, y)` must throw an exception if and only if `compare(y, x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `((compare(x, y)>0) && (compare(y, z)>0))` implies `compare(x, z)>0`.

Finally, the implementer must ensure that `compare(x, y)==0` implies that `sgn(compare(x, z))==sgn(compare(y, z))` for all z.

It is generally the case, but not strictly required that `(compare(x, y)==0) == (x == y)`. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

Parameters

o1 - the first object to be compared

o2 - the second object to be compared

Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implements `decaf::util::Comparator< E >` (p.1128).

6.472.3.2 `template<typename E > virtual bool decaf::util::comparators::Less< E >::operator() (const E & left, const E & right) const` [inline, virtual]

Implementation of the Binary function interface as a means of allowing a **Comparator** (p.1127) to be passed to an STL **Map** (p.2305) for use as the sorting criteria.

Parameters

left - the Left hand side operand.

right - the Right hand side operand.

Returns

true if the vale of left is less than the value of right.

Implements **decaf::util::Comparator**< E > (p. 1129).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/comparators/Less.h`

6.473 **std::less**< **decaf::lang::ArrayPointer**< T > > Struct Template Reference

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

Inheritance diagram for **std::less**< **decaf::lang::ArrayPointer**< T > >:

Public Member Functions

- **bool operator()** (const **decaf::lang::ArrayPointer**< T > &left, const **decaf::lang::ArrayPointer**< T > &right) const

6.473.1 Detailed Description

```
template<typename T> struct std::less< decaf::lang::ArrayPointer< T > >
```

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

6.473.2 Member Function Documentation

- 6.473.2.1** `template<typename T > bool std::less< decaf::lang::ArrayPointer< T > >::operator() (const decaf::lang::ArrayPointer< T > & left, const decaf::lang::ArrayPointer< T > & right) const [inline]`

References **decaf::lang::ArrayPointer**< T, REFCOUNTER >::get().

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/ArrayPointer.h`

6.474 std::less< decaf::lang::Pointer< T > > Struct Template Reference

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

```
#include <src/main/decaf/lang/Pointer.h>
```

Inheritance diagram for std::less< decaf::lang::Pointer< T > >:

Public Member Functions

- bool **operator()** (const **decaf::lang::Pointer**< T > &left, const **decaf::lang::Pointer**< T > &right) const

6.474.1 Detailed Description

```
template<typename T> struct std::less< decaf::lang::Pointer< T > >
```

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

6.474.2 Member Function Documentation

6.474.2.1 template<typename T > bool std::less< decaf::lang::Pointer< T > >::operator() (const decaf::lang::Pointer< T > & *left*, const decaf::lang::Pointer< T > & *right*) const [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.475 decaf::util::logging::Level Class Reference

The **Level** (p.2185) class defines a set of standard logging levels that can be used to control logging output.

```
#include <src/main/decaf/util/logging/Level.h>
```

Inheritance diagram for decaf::util::logging::Level:

Public Member Functions

- virtual ~**Level** ()
- int **intValue** () const
- std::string **getName** () const
- std::string **toString** () const

- virtual int **compareTo** (const **Level** &value) const
- virtual bool **equals** (const **Level** &value) const
- virtual bool **operator==** (const **Level** &value) const
- virtual bool **operator<** (const **Level** &value) const

Static Public Member Functions

- static **Level** **parse** (const std::string &name) throw (decaf::lang::exceptions::IllegalArgumentException)
*Parse a level name string into a **Level** (p. 2185).*

Static Public Attributes

- static const **Level** **INHERIT**
*NULL is a special level that indicates that the **Logger** (p. 2237) should get its **Level** (p. 2185) from its parent **Logger** (p. 2237), the value is initialized as zero.*
- static const **Level** **OFF**
OFF is a special level that can be used to turn off logging.
- static const **Level** **SEVERE**
SEVERE is a message level indicating a serious failure.
- static const **Level** **WARNING**
WARNING is a message level indicating a potential problem.
- static const **Level** **INFO**
INFO is a message level for informational messages.
- static const **Level** **DEBUG**
DEBUG is a level for more verbose informative messages.
- static const **Level** **CONFIG**
CONFIG is a message level for static configuration messages.
- static const **Level** **FINE**
FINE is a message level providing tracing information.
- static const **Level** **FINER**
FINER indicates a fairly detailed tracing message.
- static const **Level** **FINEST**
FINEST indicates a highly detailed tracing message.
- static const **Level** **ALL**
ALL indicates that all messages should be logged.

Protected Member Functions

- **Level** (const std::string &name, int value)

*Create a named **Level** (p. 2185) with a given integer value.*

6.475.1 Detailed Description

The **Level** (p. 2185) class defines a set of standard logging levels that can be used to control logging output. The logging **Level** (p. 2185) objects are ordered and are specified by ordered integers. Enabling logging at a given level also enables logging at all higher levels.

Clients should normally use the predefined **Level** (p. 2185) constants such as **Level.SEVERE** (p. 2190).

The levels in descending order are:

* SEVERE (highest value) * WARNING * INFO * DEBUG * CONFIG * FINE * FINER * FINEST (lowest value)

In addition there is a level OFF that can be used to turn off logging, and a level ALL that can be used to enable logging of all messages.

It is possible for third parties to define additional logging levels by subclassing **Level** (p. 2185). In such cases subclasses should take care to chose unique integer level values.

Since

1.0

6.475.2 Constructor & Destructor Documentation

6.475.2.1 decaf::util::logging::Level::Level (const std::string & name, int value) [protected]

Create a named **Level** (p. 2185) with a given integer value.

Parameters

name Name of the level, e.g. SEVERE

value Unique integer value of this level, e.g. 100

6.475.2.2 `virtual decaf::util::logging::Level::~~Level () [virtual]`

6.475.3 Member Function Documentation

6.475.3.1 `virtual int decaf::util::logging::Level::compareTo (const Level & value) const [virtual]`

6.475.3.2 `virtual bool decaf::util::logging::Level::equals (const Level & value) const [virtual]`

6.475.3.3 `std::string decaf::util::logging::Level::getName () const [inline]`

Returns

the name of this **Level** (p.2185) instance.

6.475.3.4 `int decaf::util::logging::Level::intValue () const [inline]`

Returns

the integer value of this level instance.

6.475.3.5 `virtual bool decaf::util::logging::Level::operator< (const Level & value) const [virtual]`

6.475.3.6 `virtual bool decaf::util::logging::Level::operator== (const Level & value) const [virtual]`

6.475.3.7 `static Level decaf::util::logging::Level::parse (const std::string & name) throw (decaf::lang::exceptions::IllegalArgumentException) [static]`

Parse a level name string into a **Level** (p.2185).

The argument string may consist of either a level name or an integer value.

For example:

```
* "SEVERE" * "1000"
```

Parameters

name - The name or int value of the desired **Level** (p.2185)

Returns

the parsed **Level** (p.2185) value, passing in a level name that is an int value that is not one of the known **Level** (p.2185) values will result in a new **Level** (p.2185) that has been initialized with that int value and name as the string form of the int.

Exceptions

IllegalArgumentException if the value is not valid, validity means that the string is either a valid int (between `Integer::MIN_VALUE` and `Integer::MAX_VALUE` or is one of the known level names.

6.475.3.8 `std::string decaf::util::logging::Level::toString () const` [inline]**Returns**

the string value of this **Level** (p. 2185), e.g. "SEVERE".

6.475.4 **Field Documentation****6.475.4.1** `const Level decaf::util::logging::Level::ALL` [static]

ALL indicates that all messages should be logged.

This level is initialized to Integer::MIN_VALUE.

6.475.4.2 `const Level decaf::util::logging::Level::CONFIG` [static]

CONFIG is a message level for static configuration messages.

CONFIG messages are intended to provide a variety of static configuration information, to assist in debugging problems that may be associated with particular configurations. For example, CONFIG message might include the CPU type, the System properties, etc. This level is initialized to 600.

6.475.4.3 `const Level decaf::util::logging::Level::DEBUG` [static]

DEBUG is a level for more verbose informative messages.

DEBUG messages are intended to provide a more detailed message intended for use by developers in tracking the behavior of a client. DEBUG messages typically contain more implementation specific information that might not be significant to end users or system admins. This level is initialized to 700.

6.475.4.4 `const Level decaf::util::logging::Level::FINE` [static]

FINE is a message level providing tracing information.

All of FINE, FINER, and FINEST are intended for relatively detailed tracing. The exact meaning of the three levels will vary between subsystems, but in general, FINEST should be used for the most detailed output, FINER for somewhat less detailed output, and FINE for the lowest volume (and most important) messages.

In general the FINE level should be used for information that will be broadly interesting to developers who do not have a specialized interest in the specific subsystem.

FINE messages might include things like minor (recoverable) failures. Issues indicating potential performance problems are also worth logging as FINE. This level is initialized to 500.

6.475.4.5 `const Level decaf::util::logging::Level::FINER` [static]

FINER indicates a fairly detailed tracing message.

By default logging calls for entering, returning, or throwing an exception are traced at this level. This level is initialized to 400.

6.475.4.6 const Level decaf::util::logging::Level::FINEST [static]

FINEST indicates a highly detailed tracing message.

This level is initialized to 300.

6.475.4.7 const Level decaf::util::logging::Level::INFO [static]

INFO is a message level for informational messages.

Typically INFO messages will be written to the console or its equivalent. So the INFO level should only be used for reasonably significant messages that will make sense to end users and system admins. This level is initialized to 800.

6.475.4.8 const Level decaf::util::logging::Level::INHERIT [static]

NULL is a special level that indicates that the **Logger** (p. 2237) should get its **Level** (p. 2185) from its parent **Logger** (p. 2237), the value is initialized as zero.

6.475.4.9 const Level decaf::util::logging::Level::OFF [static]

OFF is a special level that can be used to turn off logging.

This level is initialized to Integer::MAX_VALUE

6.475.4.10 const Level decaf::util::logging::Level::SEVERE [static]

SEVERE is a message level indicating a serious failure.

In general SEVERE messages should describe events that are of considerable importance and which will prevent normal program execution. They should be reasonably intelligible to end users and to system administrators. This level is initialized to 1000.

6.475.4.11 const Level decaf::util::logging::Level::WARNING [static]

WARNING is a message level indicating a potential problem.

In general WARNING messages should describe events that will be of interest to end users or system managers, or which indicate potential problems. This level is initialized to 900.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Level.h**

6.476 decaf::util::List< E > Class Template Reference

An ordered collection (also known as a sequence).

```
#include <src/main/decaf/util/List.h>
```

Inheritance diagram for decaf::util::List< E >:

Public Member Functions

- **List** ()
- virtual **~List** ()
- virtual **ListIterator**< E > * **listIterator** ()=0
- virtual **ListIterator**< E > * **listIterator** () const =0
- virtual **ListIterator**< E > * **listIterator** (std::size_t index)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual **ListIterator**< E > * **listIterator** (std::size_t index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual std::size_t **indexOf** (const E &value)=0 throw (decaf::lang::exceptions::NoSuchElementException)
Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
- virtual std::size_t **lastIndexOf** (const E &value)=0 throw (decaf::lang::exceptions::NoSuchElementException)
Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
- virtual E **get** (std::size_t index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Gets the element contained at position passed.
- virtual E **set** (std::size_t index, const E &element)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Replaces the element at the specified position in this list with the specified element.
- virtual void **add** (std::size_t index, const E &element)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)
Inserts the specified element at the specified position in this list.
- virtual bool **addAll** (std::size_t index, const **Collection**< E > &source)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)
Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
- virtual E **remove** (std::size_t index)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)
Removes the element at the specified position in this list.

6.476.1 Detailed Description

template<typename E> class decaf::util::List< E >

An ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements `e1` and `e2` such that `e1.equals(e2)`, and they typically allow multiple null elements if they allow null elements at all. It is not inconceivable that someone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

6.476.2 Constructor & Destructor Documentation

6.476.2.1 `template<typename E> decaf::util::List< E >::List () [inline]`

6.476.2.2 `template<typename E> virtual decaf::util::List< E >::~~List () [inline, virtual]`

6.476.3 Member Function Documentation

6.476.3.1 `template<typename E> virtual void decaf::util::List< E >::add (std::size_t index, const E & element) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

index - index at which the specified element is to be inserted

element - element to be inserted

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implemented in `decaf::util::StlList< E >` (p. 3363), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3363), `decaf::util::StlList< CompositeTask * >` (p. 3363), `decaf::util::StlList< URI >` (p. 3363), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3363), `decaf::util::StlList< PrimitiveValueNode >` (p. 3363), `decaf::util::StlList< Pointer< Command > >` (p. 3363), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3363), `decaf::util::StlList< cms::MessageProducer * >` (p. 3363), `decaf::util::StlList< cms::Destination * >` (p. 3363), `decaf::util::StlList< cms::Session * >` (p. 3363), and `decaf::util::StlList< cms::Connection * >` (p. 3363).

6.476.3.2 `template<typename E> virtual bool decaf::util::List< E >::addAll (std::size_t index, const Collection< E > & source) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are

returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

index The index at which to insert the first element from the specified collection
source The **Collection** (p. 1097) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBoundsException - if the index is greater than size
UnsupportedOperationException - If the collection is non-modifiable.

Implemented in `decaf::util::StlList< E >` (p. 3364), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3364), `decaf::util::StlList< CompositeTask * >` (p. 3364), `decaf::util::StlList< URI >` (p. 3364), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3364), `decaf::util::StlList< PrimitiveValueNode >` (p. 3364), `decaf::util::StlList< Pointer< Command > >` (p. 3364), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3364), `decaf::util::StlList< cms::MessageProducer * >` (p. 3364), `decaf::util::StlList< cms::Destination * >` (p. 3364), `decaf::util::StlList< cms::Session * >` (p. 3364), and `decaf::util::StlList< cms::Connection * >` (p. 3364).

6.476.3.3 `template<typename E> virtual E decaf::util::List< E >::get (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Gets the element contained at position passed.

Parameters

index - position to get

Returns

value at index

Implemented in `decaf::util::StlList< E >` (p. 3365), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3365), `decaf::util::StlList< CompositeTask * >` (p. 3365), `decaf::util::StlList< URI >` (p. 3365), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3365), `decaf::util::StlList< PrimitiveValueNode >` (p. 3365), `decaf::util::StlList< Pointer< Command > >` (p. 3365), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3365), `decaf::util::StlList< cms::MessageProducer * >` (p. 3365), `decaf::util::StlList< cms::Destination * >` (p. 3365), `decaf::util::StlList< cms::Session * >` (p. 3365), and `decaf::util::StlList< cms::Connection * >` (p. 3365).

6.476.3.4 `template<typename E> virtual std::size_t decaf::util::List< E >::indexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NoSuchElementException if value is not in the list

Implemented in `decaf::util::StlList< E >` (p. 3366), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3366), `decaf::util::StlList< CompositeTask * >` (p. 3366), `decaf::util::StlList< URI >` (p. 3366), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3366), `decaf::util::StlList< PrimitiveValueNode >` (p. 3366), `decaf::util::StlList< Pointer< Command > >` (p. 3366), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3366), `decaf::util::StlList< cms::MessageProducer * >` (p. 3366), `decaf::util::StlList< cms::Destination * >` (p. 3366), `decaf::util::StlList< cms::Session * >` (p. 3366), and `decaf::util::StlList< cms::Connection * >` (p. 3366).

6.476.3.5 `template<typename E> virtual size_t decaf::util::List< E >::lastIndexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

NoSuchElementException if value is not in the list

Implemented in `decaf::util::StlList< E >` (p. 3366), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3366), `decaf::util::StlList< CompositeTask * >` (p. 3366), `decaf::util::StlList< URI >` (p. 3366), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3366), `decaf::util::StlList< PrimitiveValueNode >` (p. 3366), `decaf::util::StlList< Pointer< Command > >` (p. 3366), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3366), `decaf::util::StlList< cms::MessageProducer * >` (p. 3366), `decaf::util::StlList< cms::Destination * >` (p. 3366), `decaf::util::StlList< cms::Session * >` (p. 3366), and `decaf::util::StlList< cms::Connection * >` (p. 3366).

6.476.3.6 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator ()` [pure virtual]

Returns

a list iterator over the elements in this list (in proper sequence).

Implemented in `decaf::util::StlList< E >` (p. 3367), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3367), `decaf::util::StlList< CompositeTask * >` (p. 3367), `decaf::util::StlList< URI >` (p. 3367), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3367), `decaf::util::StlList< PrimitiveValueNode >` (p. 3367), `decaf::util::StlList< Pointer< Command > >` (p. 3367), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3367), `decaf::util::StlList< cms::MessageProducer * >` (p. 3367), `decaf::util::StlList< cms::Destination * >` (p. 3367), `decaf::util::StlList< cms::Session * >` (p. 3367), and `decaf::util::StlList< cms::Connection * >` (p. 3367).

6.476.3.7 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (std::size_t index) const` throw (`decaf::lang::exceptions::IndexOutOfBoundsException`) [pure virtual]

Implemented in `decaf::util::StlList< E >` (p. 3367), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3367), `decaf::util::StlList< CompositeTask * >` (p. 3367), `decaf::util::StlList< URI >` (p. 3367), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3367), `decaf::util::StlList< PrimitiveValueNode >` (p. 3367), `decaf::util::StlList< Pointer< Command > >` (p. 3367), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3367), `decaf::util::StlList< cms::MessageProducer * >` (p. 3367), `decaf::util::StlList< cms::Destination * >` (p. 3367), `decaf::util::StlList< cms::Session * >` (p. 3367), and `decaf::util::StlList< cms::Connection * >` (p. 3367).

6.476.3.8 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator () const` [pure virtual]

Implemented in `decaf::util::StlList< E >` (p. 3367), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3367), `decaf::util::StlList< CompositeTask * >` (p. 3367), `decaf::util::StlList< URI >` (p. 3367), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3367), `decaf::util::StlList< PrimitiveValueNode >` (p. 3367), `decaf::util::StlList< Pointer< Command > >` (p. 3367), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3367), `decaf::util::StlList< cms::MessageProducer * >` (p. 3367), `decaf::util::StlList< cms::Destination * >` (p. 3367), `decaf::util::StlList< cms::Session * >` (p. 3367), and `decaf::util::StlList< cms::Connection * >` (p. 3367).

6.476.3.9 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (std::size_t index)` throw (`decaf::lang::exceptions::IndexOutOfBoundsException`) [pure virtual]

Parameters

index index of first element to be returned from the list iterator (by a call to the next method).

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial

call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

IndexOutOfBoundsException if the index is out of range (`index < 0 || index > size()` (p. 1106))

Implemented in `decaf::util::StlList< E >` (p. 3367), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3367), `decaf::util::StlList< CompositeTask * >` (p. 3367), `decaf::util::StlList< URI >` (p. 3367), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3367), `decaf::util::StlList< PrimitiveValueNode >` (p. 3367), `decaf::util::StlList< Pointer< Command > >` (p. 3367), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3367), `decaf::util::StlList< cms::MessageProducer * >` (p. 3367), `decaf::util::StlList< cms::Destination * >` (p. 3367), `decaf::util::StlList< cms::Session * >` (p. 3367), and `decaf::util::StlList< cms::Connection * >` (p. 3367).

```
6.476.3.10  template<typename E> virtual E decaf::util::List<
              E >::remove ( std::size_t index ) throw (
              decaf::lang::exceptions::UnsupportedOperationException,
              decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]
```

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

index - the index of the element to be removed

Returns

the element previously at the specified position

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implemented in `decaf::util::StlList< E >` (p. 3368), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3368), `decaf::util::StlList< CompositeTask * >` (p. 3368), `decaf::util::StlList< URI >` (p. 3368), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3368), `decaf::util::StlList< PrimitiveValueNode >` (p. 3368), `decaf::util::StlList< Pointer< Command > >` (p. 3368), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3368), `decaf::util::StlList< cms::MessageProducer * >` (p. 3368), `decaf::util::StlList< cms::Destination * >` (p. 3368), `decaf::util::StlList< cms::Session * >` (p. 3368), and `decaf::util::StlList< cms::Connection * >` (p. 3368).

```
6.476.3.11  template<typename E> virtual E decaf::util::List< E >::set
              ( std::size_t index, const E & element ) throw (
              decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]
```

Replaces the element at the specified position in this list with the specified element.

Parameters

index - index of the element to replace

element - element to be stored at the specified position

Returns

the element previously at the specified position

Exceptions

IndexOutOfBoundsException - if the index is greater than size

Implemented in `decaf::util::StlList< E >` (p. 3369), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3369), `decaf::util::StlList< CompositeTask * >` (p. 3369), `decaf::util::StlList< URI >` (p. 3369), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3369), `decaf::util::StlList< PrimitiveValueNode >` (p. 3369), `decaf::util::StlList< Pointer< Command > >` (p. 3369), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3369), `decaf::util::StlList< cms::MessageProducer * >` (p. 3369), `decaf::util::StlList< cms::Destination * >` (p. 3369), `decaf::util::StlList< cms::Session * >` (p. 3369), and `decaf::util::StlList< cms::Connection * >` (p. 3369).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/List.h`

6.477 decaf::util::ListIterator< E > Class Template Reference

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

```
#include <src/main/decaf/util/ListIterator.h>
```

Inheritance diagram for `decaf::util::ListIterator< E >`:

Public Member Functions

- virtual `~ListIterator ()`
- virtual void `add (const E &e)=0` throw (`decaf::lang::exceptions::UnsupportedOperationException`, `decaf::lang::exceptions::IllegalArgumentException`)
Inserts the specified element into the list (optional operation).
- virtual void `set (const E &e)=0` throw (`decaf::lang::exceptions::UnsupportedOperationException`, `decaf::lang::exceptions::IllegalArgumentException`, `decaf::lang::exceptions::IllegalStateException`)
Replaces the last element returned by next or previous with the specified element (optional operation).
- virtual bool `hasPrevious ()` const =0
Returns true if this list iterator has more elements when traversing the list in the reverse direction.

- virtual E **previous** ()=0
Returns the previous element in the list.
- virtual int **nextIndex** () const =0
Returns the index of the element that would be returned by a subsequent call to next.
- virtual int **previousIndex** () const =0
Returns the index of the element that would be returned by a subsequent call to previous.

6.477.1 Detailed Description

template<typename E> class decaf::util::ListIterator< E >

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list. Note that the **remove()** (p. 2014) and **set(Object)** methods are not defined in terms of the cursor position; they are defined to operate on the last element returned by a call to **next()** (p. 2013) or **previous()** (p. 2199).

6.477.2 Constructor & Destructor Documentation

6.477.2.1 template<typename E > virtual decaf::util::ListIterator< E >::~ListIterator () [inline, virtual]

6.477.3 Member Function Documentation

6.477.3.1 template<typename E > virtual void decaf::util::ListIterator< E >::add (const E & e) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException) [pure virtual]

Inserts the specified element into the list (optional operation).

The element is inserted immediately before the next element that would be returned by next, if any, and after the next element that would be returned by previous, if any. (If the list contains no elements, the new element becomes the sole element on the list.) The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to previous would return the new element. (This call increases by one the value that would be returned by a call to nextIndex or previousIndex.)

Parameters

e - the element to insert.

Exceptions

UnsupportedOperationException - if the add method is not supported by this list iterator.

IllegalArgumentException - if some aspect of this element prevents it from being added to this list.

6.477.3.2 `template<typename E > virtual bool decaf::util::ListIterator< E >::hasPrevious () const [pure virtual]`

Returns true if this list iterator has more elements when traversing the list in the reverse direction. (In other words, returns true if previous would return an element rather than throwing an exception.)

Returns

true if the list iterator has more elements when traversing the list in the reverse direction.

6.477.3.3 `template<typename E > virtual int decaf::util::ListIterator< E >::nextIndex () const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to next. (Returns list size if the list iterator is at the end of the list.)

Returns

the index of the element that would be returned by a subsequent call to next, or list size if list iterator is at end of list.

6.477.3.4 `template<typename E > virtual E decaf::util::ListIterator< E >::previous () [pure virtual]`

Returns the previous element in the list.

This method may be called repeatedly to iterate through the list backwards, or intermixed with calls to next to go back and forth. (Note that alternating calls to next and previous will return the same element repeatedly.)

Returns

the previous element in the list.

Exceptions

NoSuchElementException - if the iteration has no previous element.

6.477.3.5 `template<typename E > virtual int decaf::util::ListIterator< E >::previousIndex () const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to previous. (Returns -1 if the list iterator is at the beginning of the list.)

Returns

the index of the element that would be returned by a subsequent call to previous, or -1 if list iterator is at beginning of list.

```

6.477.3.6  template<typename E > virtual void decaf::util::ListIterator<
             E >::set (  const E &  e  ) throw ( de-
             caf::lang::exceptions::UnsupportedOperationException,
             decaf::lang::exceptions::IllegalArgumentException,
             decaf::lang::exceptions::IllegalStateException ) [pure virtual]

```

Replaces the last element returned by next or previous with the specified element (optional operation).

This call can be made only if neither **ListIterator.remove** (p. 2014) nor **ListIterator.add** (p. 2198) have been called after the last call to next or previous.

Parameters

e - the element with which to replace the last element returned by next or previous.

Exceptions

UnsupportedOperationException - if the add method is not supported by this list iterator.

IllegalArgumentException - if some aspect of this element prevents it from being added to this list.

IllegalStateException - if neither next nor previous have been called, or remove or add have been called after the last call to next or previous.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**ListIterator.h**

6.478 activemq::commands::LocalTransactionId Class Reference

```
#include <src/main/activemq/commands/LocalTransactionId.h>
```

Inheritance diagram for activemq::commands::LocalTransactionId:

Public Types

- typedef decaf::lang::PointerComparator< LocalTransactionId > COMPARATOR

Public Member Functions

- LocalTransactionId ()
- LocalTransactionId (const LocalTransactionId &other)
- virtual ~LocalTransactionId ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual LocalTransactionId * **cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

- virtual long long **getValue** () const

- virtual void **setValue** (long long value)

- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const

- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()

- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)

- virtual int **compareTo** (const **LocalTransactionId** &value) const

- virtual bool **equals** (const **LocalTransactionId** &value) const

- virtual bool **operator==** (const **LocalTransactionId** &value) const

- virtual bool **operator<** (const **LocalTransactionId** &value) const

- **LocalTransactionId** & **operator=** (const **LocalTransactionId** &other)

Static Public Attributes

- static const unsigned char **ID_LOCALTRANSACTIONID** = 111

Protected Attributes

- long long value
- **Pointer**< **ConnectionId** > connectionId

6.478.1 Member Typedef Documentation

6.478.1.1 `typedef decaf::lang::PointerComparator<LocalTransactionId>
activemq::commands::LocalTransactionId::COMPARATOR`

6.478.2 Constructor & Destructor Documentation

6.478.2.1 `activemq::commands::LocalTransactionId::LocalTransactionId ()`

6.478.2.2 `activemq::commands::LocalTransactionId::LocalTransactionId (const
LocalTransactionId & other)`

6.478.2.3 `virtual activemq::commands::LocalTransactionId::~~LocalTransactionId ()
[virtual]`

6.478.3 Member Function Documentation

6.478.3.1 `virtual LocalTransactionId* ac-
tivemq::commands::LocalTransactionId::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.478.3.2 `virtual int activemq::commands::LocalTransactionId::compareTo (const
LocalTransactionId & value) const [virtual]`

6.478.3.3 `virtual void activemq::commands::LocalTransactionId::copyDataStructure
(const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.478.3.4 `virtual bool activemq::commands::LocalTransactionId::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.478.3.5 virtual bool activemq::commands::LocalTransactionId::equals (const LocalTransactionId & *value*) const [virtual]

6.478.3.6 virtual const Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId () const [virtual]

6.478.3.7 virtual Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId () [virtual]

6.478.3.8 virtual unsigned char activemq::commands::LocalTransactionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

6.478.3.9 virtual long long activemq::commands::LocalTransactionId::getValue () const [virtual]

6.478.3.10 virtual bool activemq::commands::LocalTransactionId::operator< (const LocalTransactionId & *value*) const [virtual]

6.478.3.11 LocalTransactionId& activemq::commands::LocalTransactionId::operator= (const LocalTransactionId & *other*)

6.478.3.12 virtual bool activemq::commands::LocalTransactionId::operator== (const LocalTransactionId & *value*) const [virtual]

6.478.3.13 virtual void activemq::commands::LocalTransactionId::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.478.3.14 virtual void activemq::commands::LocalTransactionId::setValue (long long *value*) [virtual]

6.478.3.15 virtual std::string activemq::commands::LocalTransactionId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.478.4 Field Documentation

- 6.478.4.1 `Pointer<ConnectionId> activemq::commands::LocalTransactionId::connectionId`
[protected]
- 6.478.4.2 `const unsigned char activemq::commands::LocalTransactionId::ID - LOCALTRANSACTIONID = 111` [static]
- 6.478.4.3 `long long activemq::commands::LocalTransactionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/LocalTransactionId.h`

6.479 `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `LocalTransactionIdMarshaller` (p. 2204).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller`:

Public Member Functions

- `LocalTransactionIdMarshaller ()`
- `virtual ~LocalTransactionIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.479.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.2204). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.479.2 Constructor & Destructor Documentation

6.479.2.1 activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller () [inline]

6.479.2.2 virtual activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller () [inline, virtual]

6.479.3 Member Function Documentation

6.479.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.479.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.479.3.3 virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3592).

6.479.3.4 virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3592).

6.479.3.5 virtual int activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3593).

```
6.479.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3593).

```
6.479.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3594).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/LocalTransactionIdMarshaller.h`

6.480 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2208).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.480.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2208). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.480.2 Constructor & Destructor Documentation

6.480.2.1 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller () [inline]

6.480.2.2 virtual activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller () [inline, virtual]

6.480.3 Member Function Documentation

6.480.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.480.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.480.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3581).

6.480.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3581).

6.480.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3582).

```
6.480.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3582).

```
6.480.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3583).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h`

6.481 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2212).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller`:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.481.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2212). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.481.2 Constructor & Destructor Documentation

6.481.2.1 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller () [inline]

6.481.2.2 virtual
 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller () [inline, virtual]

6.481.3 Member Function Documentation

6.481.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.481.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.481.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3585).

6.481.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3585).

6.481.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3586).

6.481.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3586).

6.481.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3587).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h`

6.482 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2216).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.482.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2216). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.482.2 Constructor & Destructor Documentation

6.482.2.1 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller () [inline]

6.482.2.2 virtual
 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller () [inline, virtual]

6.482.3 Member Function Documentation

6.482.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.482.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.482.3.3 virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3574).

6.482.3.4 virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3574).

6.482.3.5 virtual int activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3575).

```
6.482.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3575).

```
6.482.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3576).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h`

6.483 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2220).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.483.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2220). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.483.2 Constructor & Destructor Documentation

6.483.2.1 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller () [inline]

6.483.2.2 virtual
 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller () [inline, virtual]

6.483.3 Member Function Documentation

6.483.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.483.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.483.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3588).

6.483.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3589).

6.483.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3589).

```
6.483.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3590).

```
6.483.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3590).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h`

6.484 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2224).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.484.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2224). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.484.2 Constructor & Destructor Documentation

6.484.2.1 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller () [inline]

6.484.2.2 virtual
 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller () [inline, virtual]

6.484.3 Member Function Documentation

6.484.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.484.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```

6.484.3.3  virtual void ac-
            tivemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseMarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
              decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3577).

```

6.484.3.4  virtual void ac-
            tivemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseUnmarsh
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn ) throw (
              decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3578).

```

6.484.3.5  virtual int ac-
            tivemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshall
            ( OpenWireFormat * wireFormat, commands::DataStructure
              * dataStructure, utils::BooleanStream * bs ) throw (
              decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3578).

```
6.484.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3579).

```
6.484.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3579).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h`

6.485 decaf::util::concurrent::Lock Class Reference

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

```
#include <src/main/decaf/util/concurrent/Lock.h>
```

Public Member Functions

- **Lock** (**Synchronizable** *object, const bool initiallyLocked=true)
Constructor - initializes the object member and locks the object if desired.
- virtual **~Lock** ()
Destructor - Unlocks the object if it is locked.
- void **lock** ()
Locks the object.
- void **unlock** ()
Unlocks the object if it is already locked, otherwise a call to this method has no effect.
- bool **isLocked** () const
Indicates whether or not the object is locked.

6.485.1 Detailed Description

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

Since

1.0

6.485.2 Constructor & Destructor Documentation

6.485.2.1 decaf::util::concurrent::Lock::Lock (Synchronizable * object, const bool initiallyLocked = true)

Constructor - initializes the object member and locks the object if desired.

Parameters

object The sync object to control

initiallyLocked If true, the object will automatically be locked.

6.485.2.2 virtual decaf::util::concurrent::Lock::~~Lock () [virtual]

Destructor - Unlocks the object if it is locked.

6.485.3 Member Function Documentation**6.485.3.1 bool decaf::util::concurrent::Lock::isLocked () const [inline]**

Indicates whether or not the object is locked.

Returns

true if the object is locked, otherwise false.

6.485.3.2 void decaf::util::concurrent::Lock::lock ()

Locks the object.

6.485.3.3 void decaf::util::concurrent::Lock::unlock ()

Unlocks the object if it is already locked, otherwise a call to this method has no effect.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Lock.h**

6.486 decaf::util::concurrent::locks::Lock Class Reference

Lock (p. 2229) implementations provide more extensive locking operations than can be obtained using synchronized statements.

```
#include <src/main/decaf/util/concurrent/locks/Lock.h>
```

Inheritance diagram for decaf::util::concurrent::locks::Lock:

Public Member Functions

- virtual **~Lock** ()
- virtual void **lock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Acquires the lock.
- virtual void **lockInterruptibly** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException)

Acquires the lock unless the current thread is interrupted.

- virtual bool **tryLock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Acquires the lock only if it is free at the time of invocation.
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException)
Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.
- virtual void **unlock** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Releases the lock.
- virtual **Condition** * **newCondition** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException)
*Returns a new **Condition** (p. 1156) instance that is bound to this **Lock** (p. 2229) instance.*

6.486.1 Detailed Description

Lock (p. 2229) implementations provide more extensive locking operations than can be obtained using synchronized statements. They allow more flexible structuring, may have quite different properties, and may support multiple associated **Condition** (p. 1156) objects.

A lock is a tool for controlling access to a shared resource by multiple threads. Commonly, a lock provides exclusive access to a shared resource: only one thread at a time can acquire the lock and all access to the shared resource requires that the lock be acquired first. However, some locks may allow concurrent access to a shared resource, such as the read lock of a **ReadWriteLock** (p. 2968).

While the scoping mechanism for synchronized statements makes it much easier to program with monitor locks, and helps avoid many common programming errors involving locks, there are occasions where you need to work with locks in a more flexible way. For example, some algorithms for traversing concurrently accessed data structures require the use of "hand-over-hand" or "chain locking": you acquire the lock of node A, then node B, then release A and acquire C, then release B and acquire D and so on. Implementations of the **Lock** (p. 2229) interface enable the use of such techniques by allowing a lock to be acquired and released in different scopes, and allowing multiple locks to be acquired and released in any order.

With this increased flexibility comes additional responsibility. The absence of block-structured locking removes the automatic release of locks that occurs with synchronized statements. In most cases, the following idiom should be used:

```
Lock (p. 2229) l = ...; l.lock(); try { // access the resource protected by this lock } catch(...) { l.unlock(); }
```

When locking and unlocking occur in different scopes, care must be taken to ensure that all code that is executed while the lock is held is protected by try-catch ensure that the lock is released when necessary.

Lock (p. 2229) implementations provide additional functionality over the use of synchronized methods and statements by providing a non-blocking attempt to acquire a lock (**tryLock**() (p. 2234)), an attempt to acquire the lock that can be interrupted (**lockInterruptibly**() (p. 2231), and an attempt to acquire the lock that can timeout (**tryLock(long, TimeUnit)**).

Note that **Lock** (p. 2229) instances are just normal objects and can themselves be used as the target in a synchronized statement.

The three forms of lock acquisition (interruptible, non-interruptible, and timed) may differ in their performance characteristics, ordering guarantees, or other implementation qualities. Further, the ability to interrupt the ongoing acquisition of a lock may not be available in a given **Lock** (p. 2229) class. Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of lock acquisition, nor is it required to support interruption of an ongoing lock acquisition. An implementation is required to clearly document the semantics and guarantees provided by each of the locking methods. It must also obey the interruption semantics as defined in this interface, to the extent that interruption of lock acquisition is supported: which is either totally, or only on method entry.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since

1.0

6.486.2 Constructor & Destructor Documentation

6.486.2.1 `virtual decaf::util::concurrent::locks::Lock::~~Lock () [inline, virtual]`

6.486.3 Member Function Documentation

6.486.3.1 `virtual void decaf::util::concurrent::locks::Lock::lock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]`

Acquires the lock.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired.

Implementation Considerations

A **Lock** (p. 2229) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 2229) implementation.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2980).

6.486.3.2 `virtual void decaf::util::concurrent::locks::Lock::lockInterruptibly () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException) [pure virtual]`

Acquires the lock unless the current thread is interrupted.

Acquires the lock if it is available and returns immediately.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread, and interruption of lock acquisition is supported.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return.

A **Lock** (p.2229) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p.2229) implementation.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implemented in `decaf::util::concurrent::locks::ReentrantLock` (p.2981).

6.486.3.3 virtual Condition* `decaf::util::concurrent::locks::Lock::newCondition`
 () throw (`decaf::lang::exceptions::RuntimeException`,
`decaf::lang::exceptions::UnsupportedOperationException`) [pure
 virtual]

Returns a new **Condition** (p.1156) instance that is bound to this **Lock** (p.2229) instance.

Before waiting on the condition the lock must be held by the current thread. A call to **Condition.await()** (p.1158) will atomically release the lock before waiting and re-acquire the lock before the wait returns.

Implementation Considerations

The exact operation of the **Condition** (p.1156) instance depends on the **Lock** (p.2229) implementation and must be documented by that implementation.

Returns

A new **Condition** (p.1156) instance for this **Lock** (p.2229) instance the caller must delete the returned **Condition** (p.1156) object when done with it.

Exceptions

RuntimeException if an error occurs while creating the **Condition** (p.1156).

UnsupportedOperationException if this **Lock** (p.2229) implementation does not support conditions

Implemented in `decaf::util::concurrent::locks::ReentrantLock` (p.2981).

6.486.3.4 `virtual bool decaf::util::concurrent::locks::Lock::tryLock`
(`long long time`, `const TimeUnit & unit`)
`throw (decaf::lang::exceptions::RuntimeException,`
`decaf::lang::exceptions::InterruptedException)` [pure virtual]

Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.

If the lock is available this method returns immediately with the value true. If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread, and interruption of lock acquisition is supported; or * The specified waiting time elapses

If the lock is acquired then the value true is returned.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return, or reporting a timeout.

A **Lock** (p.2229) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an (unchecked) exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p.2229) implementation.

Parameters

time the maximum time to wait for the lock

unit the time unit of the time argument

Returns

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

Exceptions

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p.2982).

6.486.3.5 `virtual bool decaf::util::concurrent::locks::Lock::tryLock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]`

Acquires the lock only if it is free at the time of invocation.

Acquires the lock if it is available and returns immediately with the value true. If the lock is not available then this method will return immediately with the value false.

A typical usage idiom for this method would be:

```
Lock (p. 2229) lock = ...; if (lock.tryLock()) { try { // manipulate protected state } catch(...) { lock.unlock(); } } else { // perform alternative actions }
```

This usage ensures that the lock is unlocked if it was acquired, and doesn't try to unlock if the lock was not acquired.

Returns

true if the lock was acquired and false otherwise

Exceptions

RuntimeException if an error occurs while acquiring the lock.

Implemented in `decaf::util::concurrent::locks::ReentrantLock` (p. 2983).

6.486.3.6 `virtual void decaf::util::concurrent::locks::Lock::unlock () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]`

Releases the lock.

Implementation Considerations

A **Lock** (p. 2229) implementation will usually impose restrictions on which thread can release a lock (typically only the holder of the lock can release it) and may throw an exception if the restriction is violated. Any restrictions and the exception type must be documented by that **Lock** (p. 2229) implementation.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

IllegalMonitorStateException if the current thread is not the owner of the lock.

Implemented in `decaf::util::concurrent::locks::ReentrantLock` (p. 2984).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/Lock.h`

6.487 decaf::util::concurrent::locks::LockSupport Class Reference

Basic thread blocking primitives for creating locks and other synchronization classes.

```
#include <src/main/decaf/util/concurrent/locks/LockSupport.h>
```

Public Member Functions

- `~LockSupport ()`

Static Public Member Functions

- static void **unpark** (`decaf::lang::Thread *thread`) throw ()
Makes available the permit for the given thread, if it was not already available.
- static void **park** () throw ()
Disables the current thread for thread scheduling purposes unless the permit is available.
- static void **parkNanos** (long long nanos) throw ()
Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.
- static void **parkUntil** (long long deadline) throw ()
Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.

6.487.1 Detailed Description

Basic thread blocking primitives for creating locks and other synchronization classes. This class associates, with each thread that uses it, a permit (in the sense of the **Semaphore** (p. 3126) class). A call to `park` will return immediately if the permit is available, consuming it in the process; otherwise it may block. A call to `unpark` makes the permit available, if it was not already available. (Unlike with Semaphores though, permits do not accumulate. There is at most one.)

Methods `park` and `unpark` provide efficient means of blocking and unblocking threads. Races between one thread invoking `park` and another thread trying to `unpark` it will preserve liveness, due to the permit. Additionally, `park` will return if the caller's thread was interrupted, and timeout versions are supported. The `park` method may also return at any other time, for "no reason", so in general must be invoked within a loop that rechecks conditions upon return. In this sense `park` serves as an optimization of a "busy wait" that does not waste as much time spinning, but must be paired with an `unpark` to be effective.

These methods are designed to be used as tools for creating higher-level synchronization utilities, and are not in themselves useful for most concurrency control applications. The `park` method is designed for use only in constructions of the form:

```
while (!canProceed()) { ... LockSupport.park(this); }
```

where neither `canProceed` nor any other actions prior to the call to `park` entail locking or blocking. Because only one permit is associated with each thread, any intermediary uses of `park` could interfere with its intended effects.

Sample Usage. Here is a sketch of a first-in-first-out non-reentrant lock class:

```
class FIFOMutex { private:  
    AtomicBoolean locked; ConcurrentLinkedQueue<Thread*> waiters;  
public:  
    void lock() {
```

```

bool wasInterrupted = false; Thread* current = Thread::currentThread(); waiters.add( current );
// Block while not first in queue or cannot acquire lock while( waiters.peek() != current ||
!locked.compareAndSet( false, true ) ) {
LockSupport.park(this); if( Thread::interrupted() ) // ignore interrupts while waiting wasInter-
rupted = true; }
waiters.remove(); if( wasInterrupted ) // reassert interrupt status on exit current.interrupt(); }
void unlock() { locked.set( false ); LockSupport.unpark (p. 2237)( waiters.peek() ); } };

```

Since

1.0

6.487.2 Constructor & Destructor Documentation

6.487.2.1 `decaf::util::concurrent::locks::LockSupport::~~LockSupport ()`

6.487.3 Member Function Documentation

6.487.3.1 `static void decaf::util::concurrent::locks::LockSupport::park () throw () [static]`

Disables the current thread for thread scheduling purposes unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- * Some other thread invokes unpark with the current thread as the target; or
- * Some other thread interrupts the current thread; or
- * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread upon return.

6.487.3.2 `static void decaf::util::concurrent::locks::LockSupport::parkNanos (long long nanos) throw () [static]`

Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

- * Some other thread invokes unpark with the current thread as the target; or
- * Some other thread interrupts the current thread; or
- * The specified waiting time elapses; or
- * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the elapsed time upon return.

Parameters

nanos the maximum number of nanoseconds to wait

6.487.3.3 static void decaf::util::concurrent::locks::LockSupport::parkUntil (long long *deadline*) throw () [static]

Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

* Some other thread invokes unpark with the current thread as the target; or * Some other thread interrupts the current thread; or * The specified deadline passes; or * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the current time upon return.

Parameters

deadline the absolute time, in milliseconds from the Epoch, to wait until

6.487.3.4 static void decaf::util::concurrent::locks::LockSupport::unpark (decaf::lang::Thread * *thread*) throw () [static]

Makes available the permit for the given thread, if it was not already available.

If the thread was blocked on park then it will unblock. Otherwise, its next call to park is guaranteed not to block. This operation is not guaranteed to have any effect at all if the given thread has not been started.

Parameters

thread the thread to unport, or NULL in which case the method has no effect.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/LockSupport.h`

6.488 decaf::util::logging::Logger Class Reference

A **Logger** (p. 2237) object is used to log messages for a specific system or application component.

```
#include <src/main/decaf/util/logging/Logger.h>
```

Public Member Functions

- virtual `~Logger ()`
- const std::string & `getName ()` const
*Gets the name of this **Logger** (p. 2237).*
- `Logger * getParent ()` const

*Gets the parent of this **Logger** (p. 2237) which will be the nearest existing **Logger** (p. 2237) in this **Loggers** namespace.*

- void **setParent** (**Logger** *parent)
*Set (p. 3220) the parent for this **Logger** (p. 2237).*
- void **addHandler** (**Handler** *handler) throw (lang::exceptions::NullPointerException)
*Add a log **Handler** (p. 1852) to receive logging messages.*
- void **removeHandler** (**Handler** *handler)
*Removes the specified **Handler** (p. 1852) from this logger, ownership of the **Handler** (p. 1852) pointer is returned to the caller.*
- const std::list< **Handler** * > & **getHandlers** () const
Gets a vector containing all the handlers that this class has been assigned to use.
- void **setFilter** (**Filter** *filter)
*Set (p. 3220) a filter to control output on this **Logger** (p. 2237).*
- const **Filter** * **getFilter** () const
*Gets the **Filter** (p. 1770) object that this class is using.*
- **Level** **getLevel** () const
*Get the log **Level** (p. 2185) that has been specified for this **Logger** (p. 2237).*
- void **setLevel** (const **Level** &level)
Set (p. 3220) the log level specifying which message levels will be logged by this logger.
- bool **getUseParentHandlers** () const
Discover whether or not this logger is sending its output to its parent logger.
- void **setUseParentHandlers** (bool value)
*Specify whether or not this logger should send its output to it's parent **Logger** (p. 2237).*
- virtual void **entering** (const std::string &blockName, const std::string &file, const int line)
Logs an Block Enter message.
- virtual void **exiting** (const std::string &blockName, const std::string &file, const int line)
Logs an Block Exit message.
- virtual void **severe** (const std::string &file, const int line, const std::string functionName, const std::string &message)
*Log a SEVERE **Level** (p. 2185) Log.*
- virtual void **warning** (const std::string &file, const int line, const std::string functionName, const std::string &message)
*Log a WARN **Level** (p. 2185) Log.*
- virtual void **info** (const std::string &file, const int line, const std::string functionName, const std::string &message)

Log a INFO Level (p. 2185) Log.

- virtual void **debug** (const std::string &file, const int line, const std::string functionName, const std::string &message)

Log a DEBUG Level (p. 2185) Log.

- virtual void **config** (const std::string &file, const int line, const std::string functionName, const std::string &message)

Log a CONFIG Level (p. 2185) Log.

- virtual void **fine** (const std::string &file, const int line, const std::string functionName, const std::string &message)

Log a FINE Level (p. 2185) Log.

- virtual void **finer** (const std::string &file, const int line, const std::string functionName, const std::string &message)

Log a FINER Level (p. 2185) Log.

- virtual void **finest** (const std::string &file, const int line, const std::string functionName, const std::string &message)

Log a FINEST Level (p. 2185) Log.

- virtual void **throwing** (const std::string &file, const int line, const std::string functionName, const decaf::lang::Throwable &thrown)

Log throwing an exception.

- virtual bool **isLoggable** (const Level &level) const

Check if a message of the given level would actually be logged by this logger.

- virtual void **log** (LogRecord &record)

Log a LogRecord (p. 2261).

- virtual void **log** (const Level &level, const std::string &message)

Log a message, with no arguments.

- virtual void **log** (const Level &levels, const std::string &file, const int line, const std::string &message,...)

Log a message, with the list of params that is formatted into the message string.

- virtual void **log** (const Level &level, const std::string &file, const int line, const std::string &message, lang::Exception &ex)

Log a message, with associated Throwable information.

Static Public Member Functions

- static Logger * **getAnonymousLogger** ()

Creates an anonymous logger.

- static Logger * **getLogger** (const std::string &name)

Find or create a logger for a named subsystem.

Protected Member Functions

- **Logger** (const std::string &name)

*Creates a new instance of the **Logger** (p. 2237) with the given name.*

6.488.1 Detailed Description

A **Logger** (p. 2237) object is used to log messages for a specific system or application component. Loggers are normally named, using a hierarchical dot-separated namespace. **Logger** (p. 2237) names can be arbitrary strings, but they should normally be based on the namespace or class name of the logged component, such as **decaf.net** (p. 130) or **org.apache.decaf**. In addition it is possible to create "anonymous" Loggers that are not stored in the **Logger** (p. 2237) namespace.

Logger (p. 2237) objects may be obtained by calls on one of the **getLogger** factory methods. These will either create a new **Logger** (p. 2237) or return a suitable existing **Logger** (p. 2237).

Logging messages will be forwarded to registered **Handler** (p. 1852) objects, which can forward the messages to a variety of destinations, including consoles, files, OS logs, etc.

Each **Logger** (p. 2237) keeps track of a "parent" **Logger** (p. 2237), which is its nearest existing ancestor in the **Logger** (p. 2237) namespace.

Each **Logger** (p. 2237) has a "Level" associated with it. This reflects a minimum **Level** (p. 2185) that this logger cares about. If a Logger's level is set to **Level::INHERIT** (p. 2190), then its effective level is inherited from its parent, which may in turn obtain it recursively from its parent, and so on up the tree.

The log level can be configured based on the properties from the logging configuration file, as described in the description of the **LogManager** (p. 2254) class. However it may also be dynamically changed by calls on the **Logger.setLevel** (p. 2247) method. If a logger's level is changed the change may also affect child loggers, since any child logger that has 'inherit' as its level will inherit its effective level from its parent.

On each logging call the **Logger** (p. 2237) initially performs a cheap check of the request level (e.g. SEVERE or FINE) against the effective log level of the logger. If the request level is lower than the log level, the logging call returns immediately.

After passing this initial (cheap) test, the **Logger** (p. 2237) will allocate a **LogRecord** (p. 2261) to describe the logging message. It will then call a **Filter** (p. 1770) (if present) to do a more detailed check on whether the record should be published. If that passes it will then publish the **LogRecord** (p. 2261) to its output Handlers. By default, loggers also publish to their parent's Handlers, recursively up the tree.

Formatting is the responsibility of the output **Handler** (p. 1852), which will typically call a **Formatter** (p. 1838).

Note that formatting need not occur synchronously. It may be delayed until a **LogRecord** (p. 2261) is actually written to an external sink.

All methods on **Logger** (p. 2237) are thread safe.

Since

1.0

6.488.2 Constructor & Destructor Documentation

6.488.2.1 decaf::util::logging::Logger::Logger (const std::string & *name*) [protected]

Creates a new instance of the **Logger** (p. 2237) with the given name.

The logger will be initially configured with a null **Level** (p. 2185) and with useParentHandlers true.

Parameters

name A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as **decaf.net** (p. 130) or org.apache.decaf. It may be empty for anonymous Loggers.

6.488.2.2 virtual decaf::util::logging::Logger::~Logger () [virtual]

6.488.3 Member Function Documentation

6.488.3.1 void decaf::util::logging::Logger::addHandler (Handler * *handler*) throw (lang::exceptions::NullPointerException)

Add a log **Handler** (p. 1852) to receive logging messages.

By default, Loggers also send their output to their parent logger. Typically the root **Logger** (p. 2237) is configured with a set of Handlers that essentially act as default handlers for all loggers.

The ownership of the given **Handler** (p. 1852) is passed to the **Logger** (p. 2237) and the **Handler** (p. 1852) will be deleted when this **Logger** (p. 2237) is destroyed unless the caller first calls removeHandler with the same pointer value as was originally given.

Parameters

handler A Logging **Handler** (p. 1852)

Exceptions

NullPointerException if the **Handler** (p. 1852) given is NULL.

6.488.3.2 virtual void decaf::util::logging::Logger::config (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a CONFIG **Level** (p. 2185) Log.

If the logger is currently enabled for the CONFIG message level then the given message is forwarded to all the registered output **Handler** (p. 1852) objects.

Parameters

file The file name where the log was generated.

line The line number where the log was generated.

functionName The name of the function that logged this.

message The message to log at this **Level** (p. 2185).

6.488.3.3 `virtual void decaf::util::logging::Logger::debug (const std::string & file,
const int line, const std::string functionName, const std::string &
message) [virtual]`

Log a **DEBUG Level** (p. 2185) Log.

If the logger is currently enabled for the **DEBUG** message level then the given message is forwarded to all the registered output **Handler** (p. 1852) objects.

Parameters

file The file name where the log was generated.

line The line number where the log was generated.

functionName The name of the function that logged this.

message The message to log at this **Level** (p. 2185).

6.488.3.4 `virtual void decaf::util::logging::Logger::entering (const std::string &
blockName, const std::string & file, const int line) [virtual]`

Logs an Block Enter message.

This is a convenience method that is used to tag a block enter, a log record with the given information is logged at the **Level::FINER** (p. 2189) log level.

Parameters

blockName The source block name, (usually `ClassName::MethodName`, or `MethodName`).

file The source file name where this method was called from.

line The source line number where this method was called from.

6.488.3.5 `virtual void decaf::util::logging::Logger::exiting (const std::string &
blockName, const std::string & file, const int line) [virtual]`

Logs an Block Exit message.

This is a convenience method that is used to tag a block enter, a log record with the given information is logged at the **Level::FINER** (p. 2189) log level.

Parameters

blockName The source block name, (usually `ClassName::MethodName`, or `MethodName`).

file The source file name where this method was called from.

line The source line number where this method was called from.

6.488.3.6 `virtual void decaf::util::logging::Logger::fine (const std::string & file,
const int line, const std::string functionName, const std::string &
message) [virtual]`

Log a **FINE Level** (p. 2185) Log.

If the logger is currently enabled for the **FINE** message level then the given message is forwarded to all the registered output **Handler** (p. 1852) objects.

Parameters

file The file name where the log was generated.
line The line number where the log was generated.
functionName The name of the function that logged this.
message The message to log at this **Level** (p.2185).

6.488.3.7 `virtual void decaf::util::logging::Logger::finer (const std::string & file,
const int line, const std::string functionName, const std::string &
message) [virtual]`

Log a FINER **Level** (p. 2185) Log.

If the logger is currently enabled for the FINER message level then the given message is forwarded to all the registered output **Handler** (p.1852) objects.

Parameters

file The file name where the log was generated.
line The line number where the log was generated.
functionName The name of the function that logged this.
message The message to log at this **Level** (p.2185).

6.488.3.8 `virtual void decaf::util::logging::Logger::finest (const std::string & file,
const int line, const std::string functionName, const std::string &
message) [virtual]`

Log a FINEST **Level** (p. 2185) Log.

If the logger is currently enabled for the FINEST message level then the given message is forwarded to all the registered output **Handler** (p.1852) objects.

Parameters

file The file name where the log was generated.
line The line number where the log was generated.
functionName The name of the function that logged this.
message The message to log at this **Level** (p.2185).

6.488.3.9 `static Logger* decaf::util::logging::Logger::getAnonymousLogger ()
[static]`

Creates an anonymous logger.

The newly created **Logger** (p.2237) is not registered in the **LogManager** (p.2254) namespace. There will be no access checks on updates to the logger. Even although the new logger is anonymous, it is configured to have the root logger ("") as its parent. This means that by default it inherits its effective level and handlers from the root logger.

The caller is responsible for destroying the returned logger.

Returns

Newly created anonymous logger

6.488.3.10 `const Filter* decaf::util::logging::Logger::getFilter () const [inline]`

Gets the **Filter** (p. 1770) object that this class is using.

Returns

the **Filter** (p. 1770) in use, (can be NULL).

6.488.3.11 `const std::list<Handler*>& decaf::util::logging::Logger::getHandlers () const`

Gets a vector containing all the handlers that this class has been assigned to use.

Returns

a list of handlers that are used by this logger

6.488.3.12 `Level decaf::util::logging::Logger::getLevel () const [inline]`

Get the log **Level** (p. 2185) that has been specified for this **Logger** (p. 2237).

The result may be the INHERIT level, which means that this logger's effective level will be inherited from its parent.

Returns

the level that is currently set

6.488.3.13 `static Logger* decaf::util::logging::Logger::getLogger (const std::string & name) [static]`

Find or create a logger for a named subsystem.

If a logger has already been created with the given name it is returned. Otherwise a new logger is created.

If a new logger is created its log level will be configured based on the **LogManager** (p. 2254) and it will be configured to also send logging output to its parent loggers Handlers. It will be registered in the **LogManager** (p. 2254) global namespace.

Parameters

name - A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as `cms` or `activemq.core.ActiveMQConnection` (p. 233)

Returns

a suitable logger.

6.488.3.14 `const std::string& decaf::util::logging::Logger::getName () const`
[inline]

Gets the name of this **Logger** (p. 2237).

Returns

logger name

6.488.3.15 `Logger* decaf::util::logging::Logger::getParent () const` [inline]

Gets the parent of this **Logger** (p. 2237) which will be the nearest existing **Logger** (p. 2237) in this Loggers namespace.

If this is the Root **Logger** (p. 2237) than this method returns NULL.

Returns

Pointer to this Loggers nearest parent **Logger** (p. 2237).

6.488.3.16 `bool decaf::util::logging::Logger::getUseParentHandlers () const`
[inline]

Discover whether or not this logger is sending its output to its parent logger.

Returns

true if using Parent Handlers

6.488.3.17 `virtual void decaf::util::logging::Logger::info (const std::string & file,
const int line, const std::string functionName, const std::string &
message)` [virtual]

Log a INFO **Level** (p. 2185) Log.

If the logger is currently enabled for the INFO message level then the given message is forwarded to all the registered output **Handler** (p. 1852) objects.

Parameters

file The file name where the log was generated.

line The line number where the log was generated.

functionName The name of the function that logged this.

message The message to log at this **Level** (p. 2185).

6.488.3.18 `virtual bool decaf::util::logging::Logger::isLoggable (const Level &
level) const` [virtual]

Check if a message of the given level would actually be logged by this logger.

This check is based on the Loggers effective level, which may be inherited from its parent.

Parameters

level - a message logging level

Returns

true if the given message level is currently being logged.

6.488.3.19 `virtual void decaf::util::logging::Logger::log (const Level & levels,
const std::string & file, const int line, const std::string & message,
...) [virtual]`

Log a message, with the list of params that is formatted into the message string.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p.1852) objects

Parameters

level the **Level** (p. 2185) to log at

file the message to log

line the line in the file

... variable length argument to format the message string.

6.488.3.20 `virtual void decaf::util::logging::Logger::log (LogRecord & record)
[virtual]`

Log a **LogRecord** (p. 2261).

All the other logging methods in this class call through this method to actually perform any logging. Subclasses can override this single method to capture all log activity.

Parameters

record - the **LogRecord** (p. 2261) to be published

6.488.3.21 `virtual void decaf::util::logging::Logger::log (const Level & level,
const std::string & message) [virtual]`

Log a message, with no arguments.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p.1852) objects

Parameters

level the **Level** (p. 2185) to log at

message the message to log

6.488.3.22 `virtual void decaf::util::logging::Logger::log (const Level & level,
const std::string & file, const int line, const std::string & message,
lang::Exception & ex) [virtual]`

Log a message, with associated Throwable information.

If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 2261) which is forwarded to all registered output handlers. Note that the thrown argument is stored in the **LogRecord** (p. 2261) thrown property, rather than the **LogRecord** (p. 2261) parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the **LogRecord** (p. 2261) message property.

Parameters

level the **Level** (p. 2185) to log at.

file File that the message was logged in.

line the line number where the message was logged at.

message the message to log.

ex the Exception to log

6.488.3.23 `void decaf::util::logging::Logger::removeHandler (Handler * handler)`

Removes the specified **Handler** (p. 1852) from this logger, ownership of the **Handler** (p. 1852) pointer is returned to the caller.

Returns silently if the given **Handler** (p. 1852) is not found.

Parameters

handler The **Handler** (p. 1852) to remove

6.488.3.24 `void decaf::util::logging::Logger::setFilter (Filter * filter)`

Set (p. 3220) a filter to control output on this **Logger** (p. 2237).

After passing the initial "level" check, the **Logger** (p. 2237) will call this **Filter** (p. 1770) to check if a log record should really be published.

The caller releases ownership of this filter to this logger

Parameters

filter The **Filter** (p. 1770) to use, (can be NULL).

6.488.3.25 `void decaf::util::logging::Logger::setLevel (const Level & level)
[inline]`

Set (p. 3220) the log level specifying which message levels will be logged by this logger.

Message levels lower than this value will be discarded. The level value **Level::OFF** (p. 2190) can be used to turn off logging.

If the new level is the **INHERIT Level** (p. 2185), it means that this node should inherit its level from its nearest ancestor with a specific (non-INHERIT) level value.

Parameters

level The new **Level** (p. 2185) value to use when logging.

6.488.3.26 `void decaf::util::logging::Logger::setParent (Logger * parent)`
[inline]

Set (p. 3220) the parent for this **Logger** (p. 2237).

This method is used by the **LogManager** (p. 2254) to update a **Logger** (p. 2237) when the namespace changes.

It should not be called from application code.

6.488.3.27 `void decaf::util::logging::Logger::setUseParentHandlers (bool value)`
[inline]

Specify whether or not this logger should send its output to it's parent **Logger** (p. 2237).

This means that any LogRecords will also be written to the parent's Handlers, and potentially to its parent, recursively up the namespace.

Parameters

value True is output is to be written to the parent

6.488.3.28 `virtual void decaf::util::logging::Logger::severe (const std::string & file, const int line, const std::string functionName, const std::string & message)` [virtual]

Log a SEVERE **Level** (p. 2185) Log.

If the logger is currently enabled for the SEVERE message level then the given message is forwarded to all the registered output **Handler** (p. 1852) objects.

Parameters

file The file name where the log was generated.

line The line number where the log was generated.

functionName The name of the function that logged this.

message The message to log at this **Level** (p. 2185).

6.488.3.29 `virtual void decaf::util::logging::Logger::throwing (const std::string & file, const int line, const std::string functionName, const decaf::lang::Throwable & thrown)` [virtual]

Log throwing an exception.

This is a convenience method to log that a method is terminating by throwing an exception. The logging is done using the FINER level.

If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 2261) which is forwarded to all registered output handlers. The LogRecord's message is set to "THROW".

Parameters

- file* The file name where the log was generated.
- line* The line number where the log was generated.
- functionName* The name of the function that logged this.
- thrown* The Throwable that will be thrown, will be cloned.

6.488.3.30 `virtual void decaf::util::logging::Logger::warning (const std::string & file, const int line, const std::string functionName, const std::string & message) [virtual]`

Log a **WARN Level** (p. 2185) Log.

If the logger is currently enabled for the **WARN** message level then the given message is forwarded to all the registered output **Handler** (p. 1852) objects.

Parameters

- file* The file name where the log was generated.
- line* The line number where the log was generated.
- functionName* The name of the function that logged this.
- message* The message to log at this **Level** (p. 2185).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Logger.h`

6.489 decaf::util::logging::LoggerHierarchy Class Reference

```
#include <src/main/decaf/util/logging/LoggerHierarchy.h>
```

Public Member Functions

- `LoggerHierarchy ()`
- `virtual ~LoggerHierarchy ()`

6.489.1 Constructor & Destructor Documentation

6.489.1.1 `decaf::util::logging::LoggerHierarchy::LoggerHierarchy ()`

6.489.1.2 `virtual decaf::util::logging::LoggerHierarchy::~~LoggerHierarchy () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/LoggerHierarchy.h`

6.490 activemq::io::LoggingInputStream Class Reference

```
#include <src/main/activemq/io/LoggingInputStream.h>
```

Inheritance diagram for `activemq::io::LoggingInputStream`:

Public Member Functions

- **LoggingInputStream** (`decaf::io::InputStream *inputStream`, `bool own=false`)
Creates a DataInputStream that uses the specified underlying InputStream.
- virtual **~LoggingInputStream** ()

Protected Member Functions

- virtual `int doReadByte ()` throw (`decaf::io::IOException`)
- virtual `int doReadArrayBounded (unsigned char *buffer, int size, int offset, int length)` throw (`decaf::io::IOException`, `decaf::lang::exceptions::IndexOutOfBoundsException`, `decaf::lang::exceptions::NullPointerException`)

6.490.1 Constructor & Destructor Documentation

6.490.1.1 `activemq::io::LoggingInputStream::LoggingInputStream (decaf::io::InputStream * inputStream, bool own = false)`

Creates a DataInputStream that uses the specified underlying InputStream.

Parameters

inputStream the InputStream instance to wrap.

own indicates if this class owns the wrapped string defaults to false.

6.490.1.2 `virtual activemq::io::LoggingInputStream::~~LoggingInputStream ()`
[virtual]

6.490.2 Member Function Documentation

6.490.2.1 `virtual int activemq::io::LoggingInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length)` throw (`decaf::io::IOException`, `decaf::lang::exceptions::IndexOutOfBoundsException`, `decaf::lang::exceptions::NullPointerException`) [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p.1774).

6.490.2.2 `virtual int activemq::io::LoggingInputStream::doReadByte () throw (decaf::io::IOException)` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p.1774).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingInputStream.h`

6.491 activemq::io::LoggingOutputStream Class Reference

OutputStream filter that just logs the data being written.

```
#include <src/main/activemq/io/LoggingOutputStream.h>
```

Inheritance diagram for `activemq::io::LoggingOutputStream`:

Public Member Functions

- `LoggingOutputStream (OutputStream *next, bool own=false)`
Constructor.
- `virtual ~LoggingOutputStream ()`

Protected Member Functions

- `virtual void doWriteByte (unsigned char c) throw (decaf::io::IOException)`
- `virtual void doWriteArrayBounded (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

6.491.1 Detailed Description

OutputStream filter that just logs the data being written.

6.491.2 Constructor & Destructor Documentation

6.491.2.1 `activemq::io::LoggingOutputStream::LoggingOutputStream (OutputStream * next, bool own = false)`

Constructor.

Parameters

next The OutputStream to wrap an write logs to.

own If true, this object will control the lifetime of the output stream that it encapsulates.

6.491.2.2 virtual `activemq::io::LoggingOutputStream::~~LoggingOutputStream ()` [virtual]

6.491.3 Member Function Documentation

6.491.3.1 virtual void `activemq::io::LoggingOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length)` throw (`decaf::io::IOException`, `decaf::lang::exceptions::NullPointerException`, `decaf::lang::exceptions::IndexOutOfBoundsException`) [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1779).

6.491.3.2 virtual void `activemq::io::LoggingOutputStream::doWriteByte (unsigned char c)` throw (`decaf::io::IOException`) [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1779).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingOutputStream.h`

6.492 `activemq::transport::logging::LoggingTransport` Class Reference

A transport filter that logs commands as they are sent/received.

```
#include <src/main/activemq/transport/logging/LoggingTransport.h>
```

Inheritance diagram for `activemq::transport::logging::LoggingTransport`:

Public Member Functions

- **LoggingTransport** (const **Pointer**< **Transport** > &next)
Constructor.
- virtual **~LoggingTransport** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (`decaf::io::IOException`, `decaf::lang::exceptions::UnsupportedOperationException`)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (`decaf::io::IOException`, `decaf::lang::exceptions::UnsupportedOperationException`)
Not supported by this class - throws an exception.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Not supported by this class - throws an exception.

6.492.1 Detailed Description

A transport filter that logs commands as they are sent/received.

6.492.2 Constructor & Destructor Documentation

6.492.2.1 **activemq::transport::logging::LoggingTransport::LoggingTransport** (const **Pointer**< **Transport** > & *next*)

Constructor.

Parameters

next - the next **Transport** (p. 3629) in the chain

6.492.2.2 virtual **activemq::transport::logging::LoggingTransport::~LoggingTransport** () [inline, virtual]

6.492.3 Member Function Documentation

6.492.3.1 virtual void **activemq::transport::logging::LoggingTransport::onCommand** (const **Pointer**< **Command** > & *command*) [virtual]

Event handler for the receipt of a command.

Parameters

command - the received command object.

Reimplemented from **activemq::transport::TransportFilter** (p. 3640).

6.492.3.2 virtual void **activemq::transport::logging::LoggingTransport::oneway** (const **Pointer**< **Command** > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from `activemq::transport::TransportFilter` (p. 3640).

6.492.3.3 `virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Not supported by this class - throws an exception.

Parameters

command the command that is sent as a request

Exceptions

UnsupportedOperationException.

Reimplemented from `activemq::transport::TransportFilter` (p. 3642).

6.492.3.4 `virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request (const Pointer< Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Not supported by this class - throws an exception.

Parameters

command the command that is sent as a request

timeout the time to wait for a response.

Exceptions

UnsupportedOperationException.

Reimplemented from `activemq::transport::TransportFilter` (p. 3641).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/logging/LoggingTransport.h`

6.493 decaf::util::logging::LogManager Class Reference

There is a single global `LogManager` (p. 2254) object that is used to maintain a set of shared state about Loggers and log services.

```
#include <src/main/decaf/util/logging/LogManager.h>
```


Public Member Functions

- virtual **~LogManager** ()
- bool **addLogger** (**Logger** *logger) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Add a named logger.
- **Logger** * **getLogger** (const std::string &name)
*Retrieves or creates a new **Logger** (p. 2237) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.*
- int **getLoggerNames** (const std::vector< std::string > &names)
*Gets a list of known **Logger** (p. 2237) Names from this Manager, new loggers added while this method is in progress are not garunteed to be in the list.*
- void **setProperties** (const util::Properties &properties)
*Sets the **Properties** (p. 2927) this **LogManager** (p. 2254) should use to configure its loggers.*
- const util::Properties & **getProperties** () const
*Gets a reference to the Logging **Properties** (p. 2927) used by this logger.*
- std::string **getProperty** (const std::string &name)
*Gets the value of a named property of this **LogManager** (p. 2254).*
- void **addPropertyChangeListener** (PropertyChangeListener *listener)
*Adds a change listener for **LogManager** (p. 2254) **Properties** (p. 2927), adding the same instance of a change event listener does nothing.*
- void **removePropertyChangeListener** (PropertyChangeListener *listener)
*Removes a properties change listener from the **LogManager** (p. 2254), if the listener is not found of the param is NULL this method returns silently.*
- void **readConfiguration** () throw (decaf::io::IOException)
Reinitialize the logging properties and reread the logging configuration.
- void **readConfiguration** (decaf::io::InputStream *stream) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
*Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in **decaf.util.Properties** (p. 2927) format.*
- void **reset** ()
Reset the logging configuration.

Static Public Member Functions

- static **LogManager** & **getLogManager** ()
*Get the global **LogManager** (p. 2254) instance.*

Protected Member Functions

- **LogManager** ()
Constructor, hidden to protect against direct instantiation.
- **LogManager** (const **LogManager** &manager)
Copy Constructor.
- void **operator=** (const **LogManager** &manager)
Assignment operator.

Friends

- class **decaf::lang::Runtime**

6.493.1 Detailed Description

There is a single global **LogManager** (p. 2254) object that is used to maintain a set of shared state about Loggers and log services. This **LogManager** (p. 2254) object:

* Manages a hierarchical namespace of **Logger** (p. 2237) objects. All named Loggers are stored in this namespace. * Manages a set of logging control properties. These are simple key-value pairs that can be used by Handlers and other logging objects to configure themselves.

The global **LogManager** (p. 2254) object can be retrieved using **LogManager::getLogManager()** (p. 2259). The **LogManager** (p. 2254) object is created during class initialization and cannot subsequently be changed.

TODO By default, the **LogManager** (p. 2254) reads its initial configuration from a properties file "lib/logging.properties" in the JRE directory. If you edit that property file you can change the default logging configuration for all uses of that JRE.

In addition, the **LogManager** (p. 2254) uses two optional system properties that allow more control over reading the initial configuration:

* "decaf.logger.config.class" * "decaf.logger.config.file"

These two properties may be set via the Preferences API, or as command line property definitions to the "java" command, or as system property definitions passed to JNI_CreateJavaVM.

If the "java.util.logging.config.class" property is set, then the property value is treated as a class name. The given class will be loaded, an object will be instantiated, and that object's constructor is responsible for reading in the initial configuration. (That object may use other system properties to control its configuration.) The alternate configuration class can use readConfiguration(InputStream) to define properties in the **LogManager** (p. 2254).

If "decaf.util.logging.config.class" property is not set, then the "decaf.util.logging.config.file" system property can be used to specify a properties file (in **decaf.util.Properties** (p. 2927) format). The initial logging configuration will be read from this file.

If neither of these properties is defined then, as described above, the **LogManager** (p. 2254) will read its initial configuration from a properties file "lib/logging.properties" in the working directory.

The properties for loggers and Handlers will have names starting with the dot-separated name for the handler or logger. ***TODO***

The global logging properties may include:

* A property "handlers". This defines a whitespace separated list of class names for handler classes to load and register as handlers on the root **Logger** (p. 2237) (the **Logger** (p. 2237) named ""). Each class name must be for a **Handler** (p. 1852) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used. * A property "<logger>.handlers". This defines a whitespace or comma separated list of class names for handlers classes to load and register as handlers to the specified logger. Each class name must be for a **Handler** (p. 1852) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used. * A property "<logger>.useParentHandlers". This defines a boolean value. By default every logger calls its parent in addition to handling the logging message itself, this often result in messages being handled by the root logger as well. When setting this property to false a **Handler** (p. 1852) needs to be configured for this logger otherwise no logging messages are delivered. * A property "config". This property is intended to allow arbitrary configuration code to be run. The property defines a whitespace separated list of class names. A new instance will be created for each named class. The default constructor of each class may execute arbitrary code to update the logging configuration, such as setting logger levels, adding handlers, adding filters, etc.

Loggers are organized into a naming hierarchy based on their dot separated names. Thus "a.b.c" is a child of "a.b", but "a.b1" and "a.b2" are peers.

All properties whose names end with ".level" are assumed to define log levels for Loggers. Thus "foo.level" defines a log level for the logger called "foo" and (recursively) for any of its children in the naming hierarchy. Log Levels are applied in the order they are defined in the properties file. Thus level settings for child nodes in the tree should come after settings for their parents. The property name ".level" can be used to set the level for the root of the tree.

All methods on the **LogManager** (p. 2254) object are multi-thread safe.

Since

1.0

6.493.2 Constructor & Destructor Documentation

6.493.2.1 virtual decaf::util::logging::LogManager::~LogManager () [virtual]

6.493.2.2 decaf::util::logging::LogManager::LogManager () [protected]

Constructor, hidden to protect against direct instantiation.

6.493.2.3 decaf::util::logging::LogManager::LogManager (const LogManager & *manager*) [protected]

Copy Constructor.

Parameters

manager the Manager to copy

6.493.3 Member Function Documentation

6.493.3.1 `bool decaf::util::logging::LogManager::addLogger (Logger * logger) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)`

Add a named logger.

This does nothing and returns false if a logger with the same name is already registered.

The **Logger** (p. 2237) factory methods call this method to register each newly created **Logger** (p. 2237).

Parameters

logger The new **Logger** (p. 2237) instance to add to this **LogManager** (p. 2254).

Exceptions

NullPointerException if logger is NULL.

IllegalArgumentException if the logger has no name.

6.493.3.2 `void decaf::util::logging::LogManager::addPropertyChangeListener (PropertyChangeListener * listener)`

Adds a change listener for **LogManager** (p. 2254) **Properties** (p. 2927), adding the same instance of a change event listener does nothing.

Parameters

listener The **PropertyChangeListener** to add (can be NULL).

6.493.3.3 `Logger* decaf::util::logging::LogManager::getLogger (const std::string & name)`

Retrieves or creates a new **Logger** (p. 2237) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.

Parameters

name The name of the **Logger** (p. 2237).

6.493.3.4 `int decaf::util::logging::LogManager::getLoggerNames (const std::vector< std::string > & names)`

Gets a list of known **Logger** (p. 2237) Names from this Manager, new loggers added while this method is in progress are not guaranteed to be in the list.

Parameters

names STL Vector to hold string logger names.

Returns

names count of how many loggers were inserted.

6.493.3.5 `static LogManager& decaf::util::logging::LogManager::getLogManager () [static]`

Get the global **LogManager** (p. 2254) instance.

Returns

A reference to the global **LogManager** (p. 2254) instance.

6.493.3.6 `const util::Properties& decaf::util::logging::LogManager::getProperties () const [inline]`

Gets a reference to the Logging **Properties** (p. 2927) used by this logger.

Returns

The **Logger** (p. 2237) **Properties** (p. 2927) Pointer

6.493.3.7 `std::string decaf::util::logging::LogManager::getProperty (const std::string & name)`

Gets the value of a named property of this **LogManager** (p. 2254).

Parameters

name The name of the Property to retrieve.

Returns

the value of the property

6.493.3.8 `void decaf::util::logging::LogManager::operator= (const LogManager & manager) [protected]`

Assignment operator.

Parameters

manager the manager to assign from

6.493.3.9 `void decaf::util::logging::LogManager::readConfiguration () throw (decaf::io::IOException)`

Reinitialize the logging properties and reread the logging configuration.

The same rules are used for locating the configuration properties as are used at startup. So normally the logging properties will be re-read from the same file that was used at startup.

Any log level definitions in the new configuration file will be applied using **Logger.setLevel()** (p. 2247), if the target **Logger** (p. 2237) exists.

A **PropertyChangeEvent** will be fired after the properties are read.

Exceptions

IOException if an I/O error occurs.

6.493.3.10 `void decaf::util::logging::LogManager::readConfiguration (decaf::io::InputStream * stream) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)`

Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in **decaf.util.Properties** (p. 2927) format.

A PropertyChangeEvent will be fired after the properties are read.

Any log level definitions in the new configuration file will be applied using **Logger.setLevel()** (p. 2247), if the target **Logger** (p. 2237) exists.

Parameters

stream The InputStream to read the **Properties** (p. 2927) from.

Exceptions

NullPointerException if stream is NULL.

IOException if an I/O error occurs.

6.493.3.11 `void decaf::util::logging::LogManager::removePropertyChangeListener (PropertyChangeListener * listener)`

Removes a properties change listener from the **LogManager** (p. 2254), if the listener is not found of the param is NULL this method returns silently.

Parameters

listener The PropertyChangeListener to remove from the listeners set.

6.493.3.12 `void decaf::util::logging::LogManager::reset ()`

Reset the logging configuration.

For all named loggers, the reset operation removes and closes all Handlers and (except for the root logger) sets the level to INHERIT. The root logger's level is set to **Level::INFO** (p. 2190).

6.493.3.13 `void decaf::util::logging::LogManager::setProperties (const util::Properties & properties)`

Sets the **Properties** (p. 2927) this **LogManager** (p. 2254) should use to configure its loggers.

Once set a properties change event is fired.

Parameters

properties Pointer to read the configuration from

6.493.4 Friends And Related Function Documentation

6.493.4.1 friend class decaf::lang::Runtime [friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/LogManager.h`

6.494 decaf::util::logging::LogRecord Class Reference

LogRecord (p. 2261) objects are used to pass logging requests between the logging framework and individual log Handlers.

```
#include <src/main/decaf/util/logging/LogRecord.h>
```

Public Member Functions

- **LogRecord** ()
- virtual **~LogRecord** ()
- **Level** **getLevel** () const
*Get **Level** (p. 2185) of this log record.*
- void **setLevel** (**Level** value)
*Set (p. 3220) the **Level** (p. 2185) of this Log Record.*
- const std::string & **getLoggerName** () const
Gets the Source Logger's Name.
- void **setLoggerName** (const std::string &loggerName)
Sets the Source Logger's Name.
- const std::string & **getSourceFile** () const
Gets the Source Log File name.
- void **setSourceFile** (const std::string &sourceFile)
Sets the Source Log File Name.
- unsigned int **getSourceLine** () const
Gets the Source Log line number.
- void **setSourceLine** (unsigned int sourceLine)
Sets the Source Log line number.
- const std::string & **getMessage** () const
Gets the Message to be Logged.
- void **setMessage** (const std::string &message)
Sets the Message to be Logged.

- `const std::string & getSourceFunction () const`
Gets the name of the function where this log was logged.
- `void setSourceFunction (const std::string &functionName)`
Sets the name of the function where this log was logged.
- `long long getTimestamp () const`
Gets the time in mills that this message was logged.
- `void setTimestamp (long long timeStamp)`
Sets the time in mills that this message was logged.
- `long long getTreadId () const`
Gets the Thread Id where this Log was created.
- `void setTreadId (long long threadId)`
Sets the Thread Id where this Log was created.
- `decaf::lang::Throwable * getThrown () const`
*Gets any Throwable associated with this **LogRecord** (p. 2261).*
- `void setThrown (decaf::lang::Throwable *thrown)`
*Sets the Throwable associated with this **LogRecord** (p. 2261), the pointer becomes the property of this instance of the **LogRecord** (p. 2261) and will be deleted when the record is destroyed.*

6.494.1 Detailed Description

LogRecord (p. 2261) objects are used to pass logging requests between the logging framework and individual log Handlers. When a **LogRecord** (p. 2261) is passed into the logging framework it logically belongs to the framework and should no longer be used or updated by the client application.

Since

1.0

6.494.2 Constructor & Destructor Documentation

6.494.2.1 `decaf::util::logging::LogRecord::LogRecord ()`

6.494.2.2 `virtual decaf::util::logging::LogRecord::~~LogRecord () [virtual]`

6.494.3 Member Function Documentation

6.494.3.1 `Level decaf::util::logging::LogRecord::getLevel () const [inline]`

Get **Level** (p. 2185) of this log record.

Returns

Level (p. 2185) enumeration value.

6.494.3.2 `const std::string& decaf::util::logging::LogRecord::getLoggerName () const [inline]`

Gets the Source Logger's Name.

Returns

the source loggers name

6.494.3.3 `const std::string& decaf::util::logging::LogRecord::getMessage () const [inline]`

Gets the Message to be Logged.

Returns

the source logger's message

6.494.3.4 `const std::string& decaf::util::logging::LogRecord::getSourceFile () const [inline]`

Gets the Source Log File name.

Returns

the source loggers name

6.494.3.5 `const std::string& decaf::util::logging::LogRecord::getSourceFunction () const [inline]`

Gets the name of the function where this log was logged.

Returns

the source logger's message

6.494.3.6 `unsigned int decaf::util::logging::LogRecord::getSourceLine () const [inline]`

Gets the Source Log line number.

Returns

the source loggers line number

6.494.3.7 `decaf::lang::Throwable* decaf::util::logging::LogRecord::getThrown () const [inline]`

Gets any Throwable associated with this **LogRecord** (p. 2261).

Returns

point to a Throwable instance or Null.

6.494.3.8 `long long decaf::util::logging::LogRecord::getTimestamp () const`
 `[inline]`

Gets the time in mills that this message was logged.

Returns

UTC time in milliseconds

6.494.3.9 `long long decaf::util::logging::LogRecord::getTreadId () const` `[inline]`

Gets the Thread Id where this Log was created.

Returns

the source loggers line number

6.494.3.10 `void decaf::util::logging::LogRecord::setLevel (Level value)` `[inline]`

Set (p. 3220) the **Level** (p. 2185) of this Log Record.

Parameters

value **Level** (p. 2185) Enumeration Value

6.494.3.11 `void decaf::util::logging::LogRecord::setLoggerName (const std::string`
 `& loggerName)` `[inline]`

Sets the Source Logger's Name.

Parameters

loggerName the source loggers name

6.494.3.12 `void decaf::util::logging::LogRecord::setMessage (const std::string &`
 `message)` `[inline]`

Sets the Message to be Logged.

Parameters

message the source loggers message

6.494.3.13 `void decaf::util::logging::LogRecord::setSourceFile (const std::string &`
 `sourceFile)` `[inline]`

Sets the Source Log File Name.

Parameters

sourceFile the source loggers name

6.494.3.14 void decaf::util::logging::LogRecord::setSourceFunction (const std::string & *functionName*) [inline]

Sets the name of the function where this log was logged.

Parameters

functionName the source of the log

6.494.3.15 void decaf::util::logging::LogRecord::setSourceLine (unsigned int *sourceLine*) [inline]

Sets the Source Log line number.

Parameters

sourceLine the source logger's line number

6.494.3.16 void decaf::util::logging::LogRecord::setThrown (decaf::lang::Throwable * *thrown*) [inline]

Sets the Throwable associated with this **LogRecord** (p. 2261), the pointer becomes the property of this instance of the **LogRecord** (p. 2261) and will be deleted when the record is destroyed.

Parameters

thrown A pointer to a Throwable that will be associated with this record.

6.494.3.17 void decaf::util::logging::LogRecord::setTimestamp (long long *timeStamp*) [inline]

Sets the time in mills that this message was logged.

Parameters

timeStamp UTC Time in Milliseconds.

6.494.3.18 void decaf::util::logging::LogRecord::setTreadId (long long *threadId*) [inline]

Sets the Thread Id where this Log was created.

Parameters

threadId the source logger's line number

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogRecord.h**

6.495 decaf::util::logging::LogWriter Class Reference

```
#include <src/main/decaf/util/logging/LogWriter.h>
```

Public Member Functions

- **LogWriter** ()
- virtual **~LogWriter** ()
- virtual void **log** (const std::string &file, const int line, const std::string &prefix, const std::string &message)
Writes a message to the output destination.
- virtual void **log** (const std::string &message)
Writes a message to the output destination.

Static Public Member Functions

- static **LogWriter** & **getInstance** ()
Get the singleton instance.
- static void **returnInstance** ()
Returns a Checked out instance of this Writer.
- static void **destroy** ()
*Forcefully Delete the Instance of this **LogWriter** (p. 2266) even if there are outstanding references.*

6.495.1 Constructor & Destructor Documentation

6.495.1.1 decaf::util::logging::LogWriter::LogWriter ()

6.495.1.2 virtual decaf::util::logging::LogWriter::~~LogWriter () [virtual]

6.495.2 Member Function Documentation

6.495.2.1 static void decaf::util::logging::LogWriter::destroy () [static]

Forcefully Delete the Instance of this **LogWriter** (p. 2266) even if there are outstanding references.

6.495.2.2 static **LogWriter**& decaf::util::logging::LogWriter::getInstance ()
[static]

Get the singleton instance.

6.495.2.3 virtual void decaf::util::logging::LogWriter::log (const std::string & *message*) [virtual]

Writes a message to the output destination.

Parameters

message

6.495.2.4 virtual void decaf::util::logging::LogWriter::log (const std::string & *file*, const int *line*, const std::string & *prefix*, const std::string & *message*) [virtual]

Writes a message to the output destination.

Parameters

file

line

prefix

message

6.495.2.5 static void decaf::util::logging::LogWriter::returnInstance () [static]

Returns a Checked out instance of this Writer.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogWriter.h**

6.496 decaf::lang::Long Class Reference

```
#include <src/main/decaf/lang/Long.h>
```

Inheritance diagram for decaf::lang::Long:

Public Member Functions

- **Long** (long long value)
- **Long** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Long** ()
- virtual int **compareTo** (const **Long** &l) const
*Compares this **Long** (p. 2267) instance with another.*
- bool **equals** (const **Long** &l) const
- virtual bool **operator==** (const **Long** &l) const
Compares equality between this object and the one passed.

- virtual bool **operator<** (const **Long** &l) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const long long &l) const
*Compares this **Long** (p. 2267) instance with another.*
- bool **equals** (const long long &l) const
- virtual bool **operator==** (const long long &l) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const long long &l) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static int **bitCount** (long long value)
Returns the number of one-bits in the two's complement binary representation of the specified int value.
- static **Long decode** (const std::string &value) throw (exceptions::NumberFormatException)
*Decodes a **String** (p. 3427) into a **Long** (p. 2267).*
- static long long **highestOneBit** (long long value)
Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

- static long long **lowestOneBit** (long long value)
Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.
- static int **numberOfLeadingZeros** (long long value)
Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.
- static int **numberOfTrailingZeros** (long long value)
Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.
- static long long **parseLong** (const std::string &value) throw (exceptions::NumberFormatException)
Parses the string argument as a signed decimal long.
- static long long **parseLong** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Long** (p. 2267) object holding the value extracted from the specified string when parsed with the radix given by the second argument.*
- static long long **reverseBytes** (long long value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.
- static long long **reverse** (long long value)
Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.
- static long long **rotateLeft** (long long value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.
- static long long **rotateRight** (long long value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.
- static int **signum** (long long value)
Returns the signum function of the specified value.
- static std::string **toString** (long long value)
*Converts the long to a **String** (p. 3427) representation.*
- static std::string **toString** (long long value, int radix)
- static std::string **toHexString** (long long value)
Returns a string representation of the integer argument as an unsigned integer in base 16.
- static std::string **toOctalString** (long long value)
Returns a string representation of the long long argument as an unsigned long long in base 8.
- static std::string **toBinaryString** (long long value)

Returns a string representation of the long long argument as an unsigned long long in base 2.

- static **Long** **valueOf** (long long value)

*Returns a **Long** (p. 2267) instance representing the specified int value.*

- static **Long** **valueOf** (const std::string &value) throw (exceptions::NumberFormatException)

*Returns a **Long** (p. 2267) object holding the value given by the specified std::string.*

- static **Long** **valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)

*Returns a **Long** (p. 2267) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const int **SIZE** = 64

The size in bits of the primitive long long type.

- static const long long **MAX_VALUE** = (long long)0x7FFFFFFFFFFFFFFFFLL

The maximum value that the primitive type can hold.

- static const long long **MIN_VALUE** = (long long)0x8000000000000000LL

The minimum value that the primitive type can hold.

6.496.1 Constructor & Destructor Documentation

6.496.1.1 decaf::lang::Long::Long (long long value)

Parameters

value - the primitive long long to wrap

6.496.1.2 decaf::lang::Long::Long (const std::string & value) throw (exceptions::NumberFormatException)

Parameters

value - the long long formatted string to wrap

Exceptions

NumberFormatException

6.496.1.3 virtual decaf::lang::Long::~~Long () [inline, virtual]

6.496.2 Member Function Documentation

6.496.2.1 static int decaf::lang::Long::bitCount (long long *value*) [static]

Returns the number of one-bits in the two's complement binary representation of the specified int value.

This function is sometimes referred to as the population count.

Parameters

value - the long long to count

Returns

the number of one-bits in the two's complement binary representation of the specified long long value.

6.496.2.2 virtual unsigned char decaf::lang::Long::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

6.496.2.3 virtual int decaf::lang::Long::compareTo (const long long & *l*) const [virtual]

Compares this **Long** (p. 2267) instance with another.

Parameters

l - the **Integer** (p. 1941) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements decaf::lang::Comparable< long long > (p. 1126).

6.496.2.4 virtual int decaf::lang::Long::compareTo (const Long & *l*) const [virtual]

Compares this **Long** (p. 2267) instance with another.

Parameters

l - the **Long** (p. 2267) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.496.2.5 `static Long decaf::lang::Long::decode (const std::string & value)
throw (exceptions::NumberFormatException) [static]`

Decodes a **String** (p. 3427) into a **Long** (p. 2267).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified **String** (p. 3427) is the minus sign. No whitespace characters are permitted in the string.

Parameters

value - The string to decode

Returns

a **Long** (p. 2267) object containing the decoded value

Exceptions

NumberFormatException if the string is not formatted correctly.

6.496.2.6 `virtual double decaf::lang::Long::doubleValue () const [inline,
virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.496.2.7 `bool decaf::lang::Long::equals (const Long & l) const [inline]`

Parameters

l - the **Long** (p. 2267) object to compare against.

Returns

true if the two **Integer** (p. 1941) Objects have the same value.

6.496.2.8 `bool decaf::lang::Long::equals (const long long & l) const` [inline, virtual]

Parameters

l - the **Long** (p. 2267) object to compare against.

Returns

true if the two **Integer** (p. 1941) Objects have the same value.

Implements **decaf::lang::Comparable**< long long > (p. 1126).

6.496.2.9 `virtual float decaf::lang::Long::floatValue () const` [inline, virtual]

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.496.2.10 `static long long decaf::lang::Long::highestOneBit (long long value)` [static]

Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

value - the long long to be inspected

Returns

an long long value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.496.2.11 `virtual int decaf::lang::Long::intValue () const` [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.496.2.12 `virtual long long decaf::lang::Long::longValue () const` [inline, virtual]

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.496.2.13 `static long long decaf::lang::Long::lowestOneBit (long long value)`
[static]

Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

value - the long long to be inspected

Returns

an long long value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.496.2.14 `static int decaf::lang::Long::numberOfLeadingZeros (long long value)`
[static]

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.

Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values x:

$\text{* floor(log2(x)) = 63 - numberOfLeadingZeros(x) * ceil(log2(x)) = 64 - numberOfLeadingZeros(x - 1)}$

Parameters

value - the long long to be inspected

Returns

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

6.496.2.15 `static int decaf::lang::Long::numberOfTrailingZeros (long long value)`
[static]

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.

Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Parameters

value - the int to be inspected

Returns

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

6.496.2.16 `virtual bool decaf::lang::Long::operator< (const long long & l) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

l - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< long long >` (p. 1127).

6.496.2.17 `virtual bool decaf::lang::Long::operator< (const Long & l) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

l - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.496.2.18 `virtual bool decaf::lang::Long::operator== (const Long & l) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

l - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.496.2.19 `virtual bool decaf::lang::Long::operator== (const long long & l)
const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

l - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< long long >` (p. 1127).

6.496.2.20 `static long long decaf::lang::Long::parseLong (const std::string &
value) throw (exceptions::NumberFormatException) [static]`

Parses the string argument as a signed decimal long.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting long value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseLong(java.lang.String, int)` method.

Note that the characters LL or ULL are not permitted to appear at the end of this string as would normally be permitted in a C++ program.

Parameters

value - **String** (p. 3427) to parse

Returns

long long value

Exceptions

NumberFormatException on invalid string value

6.496.2.21 `static long long decaf::lang::Long::parseLong (const std::string &
value, int radix) throw (exceptions::NumberFormatException)
[static]`

Returns a **Long** (p. 2267) object holding the value extracted from the specified string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed long in the radix specified by the second argument, exactly as if the arguments were given to the `parseLong(std::string, int)` method. The result is a **Long** (p. 2267) object that represents the long long value specified by the string.

Parameters

value - **String** (p. 3427) to parse

radix - the base encoding of the string

Returns

long long value

Exceptions

NumberFormatException on invalid string value

6.496.2.22 static long long decaf::lang::Long::reverse (long long *value*) [static]

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.

Parameters

value - the value whose bits are to be reversed

Returns

the reversed bits long long.

6.496.2.23 static long long decaf::lang::Long::reverseBytes (long long *value*) [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.

Parameters

value - the long long whose bytes we are to reverse

Returns

the reversed long long.

6.496.2.24 static long long decaf::lang::Long::rotateLeft (long long *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

(Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: rotateLeft(val, -distance) == rotateRight(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F).

Parameters

value - the long long to be inspected

distance - the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

6.496.2.25 `static long long decaf::lang::Long::rotateRight (long long value, int distance) [static]`

Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

(Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: `rotateRight(val, -distance) == rotateLeft(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateRight(val, distance) == rotateRight(val, distance & 0x1F)`.

Parameters

value - the long long to be inspected
distance - the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

6.496.2.26 `virtual short decaf::lang::Long::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

6.496.2.27 `static int decaf::lang::Long::signum (long long value) [static]`

Returns the signum function of the specified value.

(The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

Parameters

value - the long long to be inspected

Returns

the signum function of the specified long long value.

6.496.2.28 `static std::string decaf::lang::Long::toBinaryString (long long value) [static]`

Returns a string representation of the long long argument as an unsigned long long in base 2.

The unsigned long long value is the argument plus 2^{32} if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with

no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The characters '0' and '1' are used as binary digits.

Parameters

value - the long long to be translated to a binary string

Returns

the unsigned long long value as a binary string

6.496.2.29 `static std::string decaf::lang::Long::toHexString (long long value)`
[static]

Returns a string representation of the integer argument as an unsigned integer in base 16.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

Parameters

value - the long long to be translated to an Octal string

Returns

the unsigned long long value as a Octal string

6.496.2.30 `static std::string decaf::lang::Long::toOctalString (long long value)`
[static]

Returns a string representation of the long long argument as an unsigned long long in base 8.

The unsigned long long value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

Parameters

value - the long long to be translated to an Octal string

Returns

the unsigned long long value as a Octal string

6.496.2.31 `static std::string decaf::lang::Long::toString (long long value)`
[static]

Converts the long to a **String** (p. 3427) representation.

Parameters

value The long to convert to a std::string.

Returns

string representation

6.496.2.32 `std::string decaf::lang::Long::toString () const`

Returns

this **Long** (p. 2267) Object as a **String** (p. 3427) Representation

6.496.2.33 `static std::string decaf::lang::Long::toString (long long value, int radix)` [static]

6.496.2.34 `static Long decaf::lang::Long::valueOf (long long value)` [inline, static]

Returns a **Long** (p. 2267) instance representing the specified int value.

Parameters

value - the long long to wrap

Returns

the new **Integer** (p. 1941) object wrapping value.

6.496.2.35 `static Long decaf::lang::Long::valueOf (const std::string & value, int radix) throw (exceptions::NumberFormatException)` [static]

Returns a **Long** (p. 2267) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed long long in the radix specified by the second argument, exactly as if the argument were given to the `parseLong(std::string, int)` method. The result is a **Long** (p. 2267) object that represents the long long value specified by the string.

Parameters

value - std::string to parse as base (radix)

radix - base of the string to parse.

Returns

new **Long** (p. 2267) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a valid long long.

6.496.2.36 `static Long decaf::lang::Long::valueOf (const std::string & value)
throw (exceptions::NumberFormatException) [static]`

Returns a **Long** (p. 2267) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal long long, exactly as if the argument were given to the `parseLong(std::string)` method. The result is a **Integer** (p. 1941) object that represents the long long value specified by the string.

Parameters

value - std::string to parse as base 10

Returns

new **Long** (p. 2267) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a decimal long long.

6.496.3 Field Documentation

6.496.3.1 `const long long decaf::lang::Long::MAX_VALUE = (long
long)0x7FFFFFFFFFFFFFFFLL [static]`

The maximum value that the primitive type can hold.

6.496.3.2 `const long long decaf::lang::Long::MIN_VALUE = (long
long)0x8000000000000000LL [static]`

The minimum value that the primitive type can hold.

6.496.3.3 `const int decaf::lang::Long::SIZE = 64 [static]`

The size in bits of the primitive long long type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Long.h`

6.497 decaf::internal::nio::LongArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/LongArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::LongArrayBuffer:

Public Member Functions

- **LongArrayBuffer** (int size, bool readOnly=false) throw (decaf::lang::exceptions::IllegalArgumentException)

*Creates a **IntArrayBuffer** (p. 1921) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **LongArrayBuffer** (long long *array, int size, int offset, int length, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

*Creates a **LongArrayBuffer** (p. 2281) object that wraps the given array.*

- **LongArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

- **LongArrayBuffer** (const LongArrayBuffer &other)

*Create a **LongArrayBuffer** (p. 2281) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual ~**LongArrayBuffer** ()
- virtual long long * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

*the array that backs this **Buffer** (p. 855).*

Exceptions

***ReadOnlyBufferException** (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.*

- virtual int **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset long longo the backing array where index zero starts.

Exceptions

***ReadOnlyBufferException** (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.*

- virtual LongBuffer * **asReadOnlyBuffer** () const

Creates a new, read-only long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow

the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only long long buffer which the caller then owns.

- virtual LongBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n + 1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

*a reference to this **LongBuffer** (p. 2291).*

Exceptions

***ReadOnlyBufferException** (p. 2966) if this buffer is read-only.*

- virtual LongBuffer * **duplicate** ()

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*a new long long **Buffer** (p. 855) which the caller owns.*

- virtual long long **get** () throw (decaf::nio::BufferUnderflowException)

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the long long at the current position.

Exceptions

***BufferUnderflowException** (p. 882) if there no more data to return.*

- virtual long long **get** (int index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

Reads the value at the given index.

Parameters

***index** The index in the **Buffer** (p. 855) where the long long is to be read.*

Returns

the long long that is located at the given index.

Exceptions

***IndexOutOfBoundsException** if index is not smaller than the buffer's limit, or index is negative.*

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual bool **isReadOnly** () const

- virtual LongBuffer & **put** (long long value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes the given long longs long longo this buffer at the current position, and then increments the position.

Parameters

value The long longs value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2966) if this buffer is read-only

- virtual LongBuffer & **put** (int index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes the given long longs long longo this buffer at the given index.

Parameters

*index The position in the **Buffer** (p. 855) to write the data*
value The long longs to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only

- virtual LongBuffer * **slice** () const

*Creates a new **LongBuffer** (p. 2291) whose content is a shared subsequence of this buffer's content.*

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the newly create **LongBuffer** (p. 2291) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)
*Sets this **LongArrayBuffer** (p. 2281) as Read-Only.*

6.497.1 Constructor & Destructor Documentation

6.497.1.1 `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (int size, bool readOnly = false) throw (decaf::lang::exceptions::IllegalArgumentException)`

Creates a **IntArrayBuffer** (p. 1921) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

size The size of the array, this is the limit we read and write to.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArgumentException if the capacity value is negative.

6.497.1.2 `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (long long * array, int size, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a **LongArrayBuffer** (p. 2281) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The actual array to wrap.

size The size of the given array.

offset The position that is this buffers start position.

length The limit of how many bytes into the array this Buffer can write.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException if buffer is NULL

IndexOutOfBoundsException if offset is greater than array capacity.

6.497.1.3 `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer`
 (`const decaf::lang::Pointer< ByteArrayAdapter > &`
`array`, `int offset`, `int length`, `bool readOnly = false`
) `throw (decaf::lang::exceptions::NullPointerException,`
`decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed `ByteArrayAdapter` and start at the given offset.

The capacity and limit of the new **LongArrayBuffer** (p.2281) will be that of the remaining capacity of the passed buffer.

Parameters

array The `ByteArrayAdapter` to wrap.

offset The position that is this buffers start position.

length The limit of how many bytes into the array this Buffer can write.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException if array is NULL

IndexOutOfBoundsException if offset + length is greater than array size.

6.497.1.4 `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (const`
`LongArrayBuffer & other)`

Create a **LongArrayBuffer** (p.2281) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayAdapter` and when changes are made to that data it is reflected in both.

Parameters

other The **LongArrayBuffer** (p.2281) this one is to mirror.

6.497.1.5 `virtual decaf::internal::nio::LongArrayBuffer::~~LongArrayBuffer ()`
`[virtual]`

6.497.2 Member Function Documentation

6.497.2.1 `virtual long long* decaf::internal::nio::LongArrayBuffer::array ()`
`throw (decaf::lang::exceptions::UnsupportedOperationException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p.855).

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 2294).

6.497.2.2 `virtual int decaf::internal::nio::LongArrayBuffer::arrayOffset ()
throw (decaf::lang::exceptions::UnsupportedOperationException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset long longo the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 2294).

6.497.2.3 `virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::asReadOnlyBuffer () const [virtual]`

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only long long buffer which the caller then owns.

Implements **decaf::nio::LongBuffer** (p. 2295).

6.497.2.4 `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::compact ()
throw (decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **LongBuffer** (p. 2291).

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::LongBuffer** (p. 2295).

6.497.2.5 **virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::duplicate ()** [virtual]

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new long long **Buffer** (p. 855) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 2296).

6.497.2.6 **virtual long long decaf::internal::nio::LongArrayBuffer::get () throw (decaf::nio::BufferUnderflowException)** [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the long long at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return.

Implements **decaf::nio::LongBuffer** (p. 2297).

6.497.2.7 `virtual long long decaf::internal::nio::LongArrayBuffer::get (int index)
const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Absolute get method.

Reads the value at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the long long is to be read.

Returns

the long long that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::LongBuffer** (p. 2296).

6.497.2.8 `virtual bool decaf::internal::nio::LongArrayBuffer::hasArray () const
[inline, virtual]`

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::LongBuffer** (p. 2298).

6.497.2.9 `virtual bool decaf::internal::nio::LongArrayBuffer::isReadOnly () const
[inline, virtual]`

6.497.2.10 `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put
(int index, long long value) throw (
lang::exceptions::IndexOutOfBoundsException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given long longs longo this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data

value The long longs to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 2298).

6.497.2.11 **virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put (long long *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)** [virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

Parameters

value The long longs value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 2298).

6.497.2.12 **virtual void decaf::internal::nio::LongArrayBuffer::setReadOnly (bool *value*)** [inline, protected, virtual]

Sets this **LongArrayBuffer** (p. 2281) as Read-Only.

Parameters

value Boolean value, true if this buffer is to be read-only, false otherwise.

6.497.2.13 **virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::slice ()**
const [virtual]

Creates a new **LongBuffer** (p. 2291) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **LongBuffer** (p. 2291) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 2300).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/LongArrayBuffer.h`

6.498 decaf::nio::LongBuffer Class Reference

This class defines four categories of operations upon long long buffers:

```
#include <src/main/decaf/nio/LongBuffer.h>
```

Inheritance diagram for decaf::nio::LongBuffer:

Public Member Functions

- virtual **~LongBuffer** ()
- virtual `std::string toString () const`
- virtual `long long * array ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)`
Returns the long long array that backs this buffer (optional operation).
- virtual `int arrayOffset ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)`
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **LongBuffer * asReadOnlyBuffer () const =0**
Creates a new, read-only long long buffer that shares this buffer's content.
- virtual **LongBuffer & compact ()=0 throw (ReadOnlyBufferException)**
Compacts this buffer.
- virtual **LongBuffer * duplicate ()=0**
Creates a new long long buffer that shares this buffer's content.
- virtual `long long get ()=0 throw (BufferUnderflowException)`
Relative get method.
- virtual `long long get (int index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
Absolute get method.
- **LongBuffer & get (std::vector< long long > buffer) throw (BufferUnderflowException)**
Relative bulk get method.

- **LongBuffer** & **get** (long long *buffer, int size, int offset, int length) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

Relative bulk get method.

- virtual bool **hasArray** () const =0

Tells whether or not this buffer is backed by an accessible long long array.

- **LongBuffer** & **put** (**LongBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the long longs remaining in the given source buffer long longo this buffer.

- **LongBuffer** & **put** (const long long *buffer, int size, int offset, int length) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

This method transfers long longs long longo this buffer from the given source array.

- **LongBuffer** & **put** (std::vector< long long > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source long longs array long longo this buffer.

- virtual **LongBuffer** & **put** (long long value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

Writes the given long longs long longo this buffer at the current position, and then increments the position.

- virtual **LongBuffer** & **put** (int index, long long value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

Writes the given long longs long longo this buffer at the given index.

- virtual **LongBuffer** * **slice** () const =0

*Creates a new **LongBuffer** (p. 2291) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **LongBuffer** &value) const

- virtual bool **equals** (const **LongBuffer** &value) const

- virtual bool **operator==** (const **LongBuffer** &value) const

- virtual bool **operator<** (const **LongBuffer** &value) const

Static Public Member Functions

- static **LongBuffer** * **allocate** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)
Allocates a new Double buffer.
- static **LongBuffer** * **wrap** (long long *array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
*Wraps the passed buffer with a new **LongBuffer** (p. 2291).*
- static **LongBuffer** * **wrap** (std::vector< long long > &buffer)
*Wraps the passed STL long long Vector in a **LongBuffer** (p. 2291).*

Protected Member Functions

- **LongBuffer** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)
*Creates a **LongBuffer** (p. 2291) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted.*

6.498.1 Detailed Description

This class defines four categories of operations upon long long buffers: o Absolute and relative get and put methods that read and write single long longs; o Relative bulk get methods that transfer contiguous sequences of long longs from this buffer long longo an array; and o Relative bulk put methods that transfer contiguous sequences of long longs from a long long array or some other long long buffer long longo this buffer o Methods for compacting, duplicating, and slicing a long long buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing long long array long longo a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.498.2 Constructor & Destructor Documentation

6.498.2.1 decaf::nio::LongBuffer::LongBuffer (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException) [protected]

Creates a **LongBuffer** (p. 2291) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity The size and limit of the **Buffer** (p. 855) in long longs.

Exceptions

IllegalArgumentException if capacity is negative.

6.498.2.2 `virtual decaf::nio::LongBuffer::~~LongBuffer () [inline, virtual]`

6.498.3 Member Function Documentation

6.498.3.1 `static LongBuffer* decaf::nio::LongBuffer::allocate (int capacity)
throw (decaf::lang::exceptions::IllegalArgumentException) [static]`

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity The size of the Double buffer in long longs.

Returns

the **LongBuffer** (p. 2291) that was allocated, caller owns.

6.498.3.2 `virtual long long* decaf::nio::LongBuffer::array () throw
(decaf::lang::exceptions::UnsupportedOperationException,
ReadOnlyBufferException) [pure virtual]`

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 855).

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2286).

6.498.3.3 `virtual int decaf::nio::LongBuffer::arrayOffset () throw
(decaf::lang::exceptions::UnsupportedOperationException,
ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset long longo the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2287).

6.498.3.4 virtual LongBuffer* decaf::nio::LongBuffer::asReadOnlyBuffer () const
[pure virtual]

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only long long buffer which the caller then owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2287).

6.498.3.5 virtual LongBuffer& decaf::nio::LongBuffer::compact () throw (
ReadOnlyBufferException) [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **LongBuffer** (p. 2291).

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2287).

6.498.3.6 `virtual int decaf::nio::LongBuffer::compareTo (const LongBuffer & value) const` [virtual]

6.498.3.7 `virtual LongBuffer* decaf::nio::LongBuffer::duplicate ()` [pure virtual]

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new long long **Buffer** (p. 855) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2288).

6.498.3.8 `virtual bool decaf::nio::LongBuffer::equals (const LongBuffer & value) const` [virtual]

6.498.3.9 `LongBuffer& decaf::nio::LongBuffer::get (std::vector< long long > buffer) throw (BufferUnderflowException)`

Relative bulk get method.

This method transfers values from this buffer long longo the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than length long longs remaining in this buffer.

6.498.3.10 `virtual long long decaf::nio::LongBuffer::get (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the long long is to be read.

Returns

the long long that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2289).

6.498.3.11 **LongBuffer& decaf::nio::LongBuffer::get (long long * *buffer*, int *size*, int *offset*, int *length*) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)**

Relative bulk get method.

This method transfers long longs from this buffer long longo the given destination array. If there are fewer long longs remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 860), then no bytes are transferred and a **BufferUnderflowException** (p. 882) is thrown.

Otherwise, this method copies length long longs from this buffer long longo the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters

buffer The pointer to an allocated long long buffer to fill.

size The size of the passed in buffer.

offset The position in the buffer to start filling.

length The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than length long longs remaining in this buffer

NullPointerException longerException if the passed buffer is null.

IndexOutOfBoundsException if the preconditions of size, offset, or length are not met.

6.498.3.12 **virtual long long decaf::nio::LongBuffer::get () throw (BufferUnderflowException) [pure virtual]**

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the long long at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2288).

6.498.3.13 `virtual bool decaf::nio::LongBuffer::hasArray () const` [pure virtual]

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in `decaf::internal::nio::LongArrayBuffer` (p. 2289).

6.498.3.14 `virtual bool decaf::nio::LongBuffer::operator< (const LongBuffer & value) const` [virtual]**6.498.3.15** `virtual bool decaf::nio::LongBuffer::operator== (const LongBuffer & value) const` [virtual]**6.498.3.16** `virtual LongBuffer& decaf::nio::LongBuffer::put (long long value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

Parameters

value The long longs value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implemented in `decaf::internal::nio::LongArrayBuffer` (p. 2290).

6.498.3.17 `virtual LongBuffer& decaf::nio::LongBuffer::put (int index, long long value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given long longs long longo this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data

value The long longs to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2289).

6.498.3.18 **LongBuffer& decaf::nio::LongBuffer::put (const long long * *buffer*, int *size*, int *offset*, int *length*) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)**

This method transfers long longs long longo this buffer from the given source array.

If there are more long longs to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 860), then no long longs are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies `length` bytes from the given array long longo this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

buffer The array from which long longs are to be read.

size The size of the buffer passed.

offset The offset within the array of the first char to be read.

length The number of long longs to be read from the given array.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer

ReadOnlyBufferException (p. 2966) if this buffer is read-only

NullPolong longerException if the passed buffer is null.

IndexOutOfBoundsException if the preconditions of size, offset, or length are not met.

6.498.3.19 **LongBuffer& decaf::nio::LongBuffer::put (std::vector< long long > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)**

This method transfers the entire content of the given source long longs array long longo this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size())`.

Parameters

buffer The buffer whose contents are copied to this **LongBuffer** (p. 2291).

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

6.498.3.20 `LongBuffer& decaf::nio::LongBuffer::put (LongBuffer & src
) throw (BufferOverflowException, ReadOnlyBufferException,
lang::exceptions::IllegalArgumentException)`

This method transfers the long longs remaining in the given source buffer long longo this buffer.

If there are more long longs remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 860), then no long longs are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies `n = src.remaining()` long longs from the given buffer long longo this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

src The buffer to take long longs from an place in this one.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer for the remaining long longs in the source buffer

IllegalArgumentException if the source buffer is this buffer

ReadOnlyBufferException (p. 2966) if this buffer is read-only

6.498.3.21 `virtual LongBuffer* decaf::nio::LongBuffer::slice () const [pure
virtual]`

Creates a new **LongBuffer** (p. 2291) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **LongBuffer** (p. 2291) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2290).

6.498.3.22 `virtual std::string decaf::nio::LongBuffer::toString () const [virtual]`**Returns**

a std::string describing this object

6.498.3.23 `static LongBuffer* decaf::nio::LongBuffer::wrap (long long * array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [static]`

Wraps the passed buffer with a new **LongBuffer** (p. 2291).

The new buffer will be backed by the given long long array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array The array that will back the new buffer.
size The size of the passed in array.
offset The offset of the subarray to be used.
length The length of the subarray to be used.

Returns

a new **LongBuffer** (p. 2291) that is backed by buffer, caller owns.

Exceptions

NullPointerException if the array pointer is NULL.
IndexOutOfBoundsException if the preconditions of size, offset, or length are not met.

6.498.3.24 `static LongBuffer* decaf::nio::LongBuffer::wrap (std::vector< long long > & buffer) [static]`

Wraps the passed STL long long Vector in a **LongBuffer** (p. 2291).

The new buffer will be backed by the given long long array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns

a new **LongBuffer** (p. 2291) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/LongBuffer.h`

6.499 activemq::util::LongSequenceGenerator Class Reference

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

```
#include <src/main/activemq/util/LongSequenceGenerator.h>
```

Public Member Functions

- **LongSequenceGenerator** ()
- virtual **~LongSequenceGenerator** ()
- long long **getNextSequenceId** ()
- long long **getLastSequenceId** ()

6.499.1 Detailed Description

This class is used to generate a sequence of long long values that are incremented each time a new value is requested. This class is thread safe so the ids can be requested in different threads safely.

6.499.2 Constructor & Destructor Documentation

6.499.2.1 `activemq::util::LongSequenceGenerator::LongSequenceGenerator ()`

6.499.2.2 `virtual
activemq::util::LongSequenceGenerator::~~LongSequenceGenerator ()
[inline, virtual]`

6.499.3 Member Function Documentation

6.499.3.1 `long long activemq::util::LongSequenceGenerator::getLastSequenceId ()`

Returns

the last id that was generated.

6.499.3.2 `long long activemq::util::LongSequenceGenerator::getNextSequenceId ()`

Returns

the next id in the sequence.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/LongSequenceGenerator.h`

6.500 decaf::net::MalformedURLException Class Reference

```
#include <src/main/decaf/net/MalformedURLException.h>
```

Inheritance diagram for decaf::net::MalformedURLException:

Public Member Functions

- **MalformedURLException** () throw ()
Default Constructor.
- **MalformedURLException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **MalformedURLException** (const MalformedURLException &ex) throw ()
Copy Constructor.
- **MalformedURLException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **MalformedURLException** (const std::exception *cause) throw ()
Constructor.
- **MalformedURLException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **MalformedURLException * clone** () const
Clones this exception.
- virtual **~MalformedURLException** () throw ()

6.500.1 Constructor & Destructor Documentation

6.500.1.1 decaf::net::MalformedURLException::MalformedURLException () throw () [inline]

Default Constructor.

6.500.1.2 `decaf::net::MalformedURLException::MalformedURLException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.500.1.3 `decaf::net::MalformedURLException::MalformedURLException (const MalformedURLException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.500.1.4 `decaf::net::MalformedURLException::MalformedURLException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.500.1.5 `decaf::net::MalformedURLException::MalformedURLException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.500.1.6 `decaf::net::MalformedURLException::MalformedURLException (const char * file, const int lineNumber, const char * msg, ...) throw ()`
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.500.1.7 `virtual decaf::net::MalformedURLException::~~MalformedURLException () throw ()` [inline, virtual]

6.500.2 Member Function Documentation

6.500.2.1 `virtual MalformedURLException* decaf::net::MalformedURLException::clone () const`
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::IOException` (p. 2005).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/MalformedURLException.h`

6.501 decaf::util::Map< K, V, COMPARATOR > Class Template Reference

Map (p. 2305) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

`#include <src/main/decaf/util/Map.h>`

Inheritance diagram for `decaf::util::Map< K, V, COMPARATOR >`:

Data Structures

- class **Entry**

Public Member Functions

- **Map** ()
Default constructor - does nothing.
- virtual **~Map** ()
- virtual bool **equals** (const **Map** &source) const =0
Comparison, equality is dependent on the method of determining if the element are equal.
- virtual void **copy** (const **Map** &source)=0
Copies the content of the source map into this map.
- virtual void **clear** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException)
Removes all keys and values from this map.
- virtual bool **containsKey** (const K &key) const =0
Indicates whether or this map contains a value for the given key.
- virtual bool **containsValue** (const V &value) const =0
Indicates whether or this map contains a value for the given value, i.e.
- virtual bool **isEmpty** () const =0
- virtual std::size_t **size** () const =0
- virtual V & **get** (const K &key)=0 throw (lang::exceptions::NoSuchElementException)
*Gets the value mapped to the specified key in the **Map** (p. 2305).*
- virtual const V & **get** (const K &key) const =0 throw (lang::exceptions::NoSuchElementException)
*Gets the value mapped to the specified key in the **Map** (p. 2305).*
- virtual void **put** (const K &key, const V &value)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)
Sets the value for the specified key.
- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)
*Stores a copy of the Mappings contained in the other **Map** (p. 2305) in this one.*
- virtual V **remove** (const K &key)=0 throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)
Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.
- virtual std::vector< K > **keySet** () const =0
*Returns a **Set** (p. 3220) view of the mappings contained in this map.*
- virtual std::vector< V > **values** () const =0

6.501.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::Map< K, V, COMPARATOR >
```

Map (p. 2305) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

6.501.2 Constructor & Destructor Documentation

6.501.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Map ()` [inline]

Default constructor - does nothing.

6.501.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::~~Map ()` [inline, virtual]

6.501.3 Member Function Documentation

6.501.3.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::clear ()` throw (`decaf::lang::exceptions::UnsupportedOperationException`) [pure virtual]

Removes all keys and values from this map.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1145), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3374), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1145), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1145), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1145), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1145), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1145), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1145), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3374), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3374), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3374), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3374),

decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3374), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3374), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3374), decaf::util::StlMap< std::string, TransportFactory * > (p. 3374), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3374), decaf::util::StlMap< int, Pointer< Command > > (p. 3374), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3374), decaf::util::StlMap< std::string, CachedProducer * > (p. 3374), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3374).

6.501.3.2 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR
>::containsKey (const K & key) const [pure virtual]`

Indicates whether or this map contains a value for the given key.

Parameters

key The key to look up.

Returns

true if this map contains the value, otherwise false.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1146), decaf::util::StlMap< K, V, COMPARATOR > (p. 3374), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1146), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1146), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1146), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1146), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1146), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1146), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3374), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3374), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3374), decaf::util::StlMap< std::string, cms::Queue * > (p. 3374), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3374), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3374), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3374), decaf::util::StlMap< std::string, TransportFactory * > (p. 3374), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3374), decaf::util::StlMap< int, Pointer< Command > > (p. 3374), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3374), decaf::util::StlMap< std::string, CachedProducer * > (p. 3374), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3374).

6.501.3.3 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR
>::containsValue (const V & value) const [pure virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters

value The Value to look up.

Returns

true if this map contains the value, otherwise false.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1146), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3374), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1146), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1146), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1146), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1146), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1146), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1146), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3374), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3374), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3374), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3374), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3374), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3374), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3374), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3374), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3374), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3374), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3374), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3374), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3374).

6.501.3.4 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR
>::copy (const Map< K, V, COMPARATOR > & source) [pure
virtual]`

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters

source The source object to copy from.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1147), and `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3375).

```
6.501.3.5  template<typename K, typename V, typename COMPARATOR =
            std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR
            >::equals ( const Map< K, V, COMPARATOR > & source ) const
            [pure virtual]
```

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters

source - `Map` (p. 2305) to compare to this one.

Returns

true if the `Map` (p. 2305) passed is equal in value to this one.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1147), and `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3375).

```
6.501.3.6  template<typename K, typename V, typename COMPARATOR =
            std::less<K>> virtual V& decaf::util::Map< K, V, COMPARATOR >::get
            ( const K & key ) throw ( lang::exceptions::NoSuchElementException )
            [pure virtual]
```

Gets the value mapped to the specified key in the `Map` (p. 2305).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters

key The search key.

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the `Map` (p. 2305).

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1147), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3376), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1147), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1147), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR`

> (p. 1147), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1147), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1147), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1147), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3376), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3376), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3376), decaf::util::StlMap< std::string, cms::Queue * > (p. 3376), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3376), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3376), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3376), decaf::util::StlMap< std::string, TransportFactory * > (p. 3376), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3376), decaf::util::StlMap< int, Pointer< Command > > (p. 3376), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3376), decaf::util::StlMap< std::string, CachedProducer * > (p. 3376), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3376).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::equals(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals().

6.501.3.7 template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual const V& decaf::util::Map< K, V,
COMPARATOR >::get (const K & *key*) const throw (
lang::exceptions::NoSuchElementException) [pure virtual]

Gets the value mapped to the specified key in the **Map** (p. 2305).

If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.

Parameters

key The search key.

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 2305).

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1148), decaf::util::StlMap< K, V, COMPARATOR > (p. 3376), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1148), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1148), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR

> (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1148), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3376), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3376), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3376), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3376), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3376), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3376), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3376), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3376), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3376), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3376), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3376), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3376), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3376).

6.501.3.8 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::isEmpty () const [pure virtual]`

Returns

if the `Map` (p. 2305) contains any element or not, `TRUE` or `FALSE`

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1148), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3377), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1148), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3377), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3377), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3377), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3377), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3377), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3377), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3377), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3377), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3377), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3377),

decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3377), decaf::util::StlMap< std::string, CachedProducer * > (p. 3377), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3377).

6.501.3.9 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual std::vector<K> decaf::util::Map< K, V,
COMPARATOR >::keySet () const [pure virtual]`

Returns a **Set** (p. 3220) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2014), **Set.remove** (p. 151), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

Returns

the entire set of keys in this map as a std::vector.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1148), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3377), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1148), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1148), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3377), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3377), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3377), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3377), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3377), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3377), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3377), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3377), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3377), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3377), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3377), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3377), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3377).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::equals()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()`.

6.501.3.10 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::put (const K & key, const V & value) throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]`

Sets the value for the specified key.

Parameters

key The target key.

value The value to be set.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1150), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3378), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1150), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1150), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1150), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1150), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1150), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1150), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3378), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3378), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3378), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3378), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3378), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3378), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3378), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3378), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3378), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3378), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3378), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3378), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3378).

6.501.3.11 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & other) throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 2305) in this one.

Parameters

other A **Map** (p. 2305) instance whose elements are to all be inserted in this **Map** (p. 2305).

Exceptions

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1150), decaf::util::StlMap< K, V, COMPARATOR > (p. 3379), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1150), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1150), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1150), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1150), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1150), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1150), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3379), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3379), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3379), decaf::util::StlMap< std::string, cms::Queue * > (p. 3379), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3379), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3379), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3379), decaf::util::StlMap< std::string, TransportFactory * > (p. 3379), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3379), decaf::util::StlMap< int, Pointer< Command > > (p. 3379), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3379), decaf::util::StlMap< std::string, CachedProducer * > (p. 3379), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3379).

```
6.501.3.12  template<typename K, typename V, typename COMPARATOR
            = std::less<K>> virtual V decaf::util::Map< K, V,
            COMPARATOR >::remove ( const K & key ) throw
            ( decaf::lang::exceptions::NoSuchElementException,
            decaf::lang::exceptions::UnsupportedOperationException ) [pure
            virtual]
```

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

key The search key.

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException if this key is not in the **Map** (p. 2305).

UnsupportedOperationException if this map is unmodifiable.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1152), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3379), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1152), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1152), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1152), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1152), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1152), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1152), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3379), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3379), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3379), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3379), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3379), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3379), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3379), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3379), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3379), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3379), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3379), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3379), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3379).

```
6.501.3.13  template<typename K, typename V, typename COMPARATOR
              = std::less<K>> virtual std::size_t decaf::util::Map< K, V,
              COMPARATOR >::size ( ) const [pure virtual]
```

Returns

The number of elements (key/value pairs) in this map.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1153), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3379), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1153), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1153), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1153), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1153), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1153),

decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1153), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3379), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3379), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3379), decaf::util::StlMap< std::string, cms::Queue * > (p. 3379), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3379), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3379), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3379), decaf::util::StlMap< std::string, TransportFactory * > (p. 3379), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3379), decaf::util::StlMap< int, Pointer< Command > > (p. 3379), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3379), decaf::util::StlMap< std::string, CachedProducer * > (p. 3379), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3379).

6.501.3.14 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual std::vector<V> decaf::util::Map< K, V,
COMPARATOR >::values () const [pure virtual]`

Returns

the entire set of values in this map as a std::vector.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1154), decaf::util::StlMap< K, V, COMPARATOR > (p. 3380), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1154), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3380), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3380), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3380), decaf::util::StlMap< std::string, cms::Queue * > (p. 3380), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3380), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3380), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3380), decaf::util::StlMap< std::string, TransportFactory * > (p. 3380), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3380), decaf::util::StlMap< int, Pointer< Command > > (p. 3380), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3380), decaf::util::StlMap< std::string, CachedProducer * > (p. 3380), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3380).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Map.h`

6.502 cms::MapMessage Class Reference

A **MapMessage** (p. 2318) object is used to send a set of name-value pairs.

```
#include <src/main/cms/MapMessage.h>
```

Inheritance diagram for cms::MapMessage:

Public Member Functions

- virtual `~MapMessage ()`
- virtual `std::vector< std::string > getMapNames () const =0 throw (CMSEException)`
*Returns an Enumeration of all the names in the **MapMessage** (p. 2318) object.*
- virtual `bool itemExists (const std::string &name) const =0 throw (CMSEException)`
*Indicates whether an item exists in this **MapMessage** (p. 2318) object.*
- virtual `bool getBoolean (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)`
Returns the Boolean value of the Specified name.
- virtual `void setBoolean (const std::string &name, bool value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)`
Sets a boolean value with the specified name into the Map.
- virtual `unsigned char getByte (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)`
Returns the Byte value of the Specified name.
- virtual `void setByte (const std::string &name, unsigned char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)`
Sets a Byte value with the specified name into the Map.
- virtual `std::vector< unsigned char > getBytes (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)`
Returns the Bytes value of the Specified name.
- virtual `void setBytes (const std::string &name, const std::vector< unsigned char > &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)`
Sets a Bytes value with the specified name into the Map.
- virtual `char getChar (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)`
Returns the Char value of the Specified name.

- virtual void **setChar** (const std::string &name, char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Char value with the specified name into the Map.
- virtual double **getDouble** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Double value of the Specified name.
- virtual void **setDouble** (const std::string &name, double value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Double value with the specified name into the Map.
- virtual float **getFloat** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Float value of the Specified name.
- virtual void **setFloat** (const std::string &name, float value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Float value with the specified name into the Map.
- virtual int **getInt** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Int value of the Specified name.
- virtual void **setInt** (const std::string &name, int value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Int value with the specified name into the Map.
- virtual long long **getLong** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Long value of the Specified name.
- virtual void **setLong** (const std::string &name, long long value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Long value with the specified name into the Map.
- virtual short **getShort** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Short value of the Specified name.
- virtual void **setShort** (const std::string &name, short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Short value with the specified name into the Map.
- virtual std::string **getString** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the String value of the Specified name.
- virtual void **setString** (const std::string &name, const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a String value with the specified name into the Map.

6.502.1 Detailed Description

A **MapMessage** (p.2318) object is used to send a set of name-value pairs. The names are String objects, and the values are primitive data types in the Java programming language. The names must have a value that is not null, and not an empty string. The entries can be accessed sequentially or randomly by name. The order of the entries is undefined. **MapMessage** (p.2318) inherits from the **Message** (p.2375) interface and adds a message body that contains a Map.

When a client receives a **MapMessage** (p.2318), it is in read-only mode. If a client attempts to write to the message at this point, a **MessageNotWriteableException** (p.2549) is thrown. To place the **MapMessage** (p.2318) back into a state where it can be read from and written to, call the `clearBody` method.

MapMessage (p.2318) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p.1074). The String-to-primitive conversions may throw a **MessageFormatException** (p.2493) if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X									X
byte		X	X		X	X				X
short			X		X	X				X
char				X						X
int					X	X				X
long						X				X
float							X	X		X
double								X		X
String	X	X	X		X	X	X	X	X	
byte[]										X

Since

1.0

6.502.2 Constructor & Destructor Documentation

6.502.2.1 `virtual cms::MapMessage::~MapMessage () [inline, virtual]`

6.502.3 Member Function Documentation

6.502.3.1 `virtual bool cms::MapMessage::getBoolean (const std::string & name) const throw (cms::MessageFormatException, cms::CMSEException) [pure virtual]`

Returns the Boolean value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.502.3.2 virtual unsigned char cms::MapMessage::getByte (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSEException) [pure virtual]

Returns the Byte value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.502.3.3 virtual std::vector<unsigned char> cms::MapMessage::getBytes (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSEException) [pure virtual]

Returns the Bytes value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.502.3.4 virtual char cms::MapMessage::getChar (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSEException) [pure virtual]

Returns the Char value of the Specified name.

Parameters

name name of the value to fetch from the map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageFormatException (p. 2493) - if this type conversion is invalid.

```

6.502.3.5 virtual double cms::MapMessage::getDouble ( const std::string & name
) const throw ( cms::MessageFormatException, cms::CMSException )
[pure virtual]

```

Returns the Double value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException (p. 1074) - if the operation fails due to an internal error.

MessageFormatException (p. 2493) - if this type conversion is invalid.

```

6.502.3.6 virtual float cms::MapMessage::getFloat ( const std::string & name
) const throw ( cms::MessageFormatException, cms::CMSException )
[pure virtual]

```

Returns the Float value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException (p. 1074) - if the operation fails due to an internal error.

MessageFormatException (p. 2493) - if this type conversion is invalid.

```

6.502.3.7 virtual int cms::MapMessage::getInt ( const std::string & name ) const
throw ( cms::MessageFormatException, cms::CMSException ) [pure
virtual]

```

Returns the Int value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException (p. 1074) - if the operation fails due to an internal error.

MessageFormatException (p. 2493) - if this type conversion is invalid.

```

6.502.3.8 virtual long long cms::MapMessage::getLong ( const std::string & name
) const throw ( cms::MessageFormatException, cms::CMSException )
[pure virtual]

```

Returns the Long value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.502.3.9 `virtual std::vector< std::string > cms::MapMessage::getMapNames ()
const throw (CMSEException) [pure virtual]`

Returns an Enumeration of all the names in the **MapMessage** (p. 2318) object.

Returns

STL Vector of String values, each of which is the name of an item in the **MapMessage** (p. 2318)

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

6.502.3.10 `virtual short cms::MapMessage::getShort (const std::string & name)
const throw (cms::MessageFormatException, cms::CMSEException)
[pure virtual]`

Returns the Short value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.502.3.11 `virtual std::string cms::MapMessage::getString (const std::string
& name) const throw (cms::MessageFormatException,
cms::CMSEException) [pure virtual]`

Returns the String value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.502.3.12 `virtual bool cms::MapMessage::itemExists (const std::string & name) const throw (CMSEException) [pure virtual]`

Indicates whether an item exists in this **MapMessage** (p. 2318) object.

Parameters

name String name of the Object in question

Returns

boolean value indicating if the name is in the map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

6.502.3.13 `virtual void cms::MapMessage::setBoolean (const std::string & name, bool value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Sets a boolean value with the specified name into the Map.

Parameters

name the name of the boolean

value the boolean value to set in the Map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p. 2375) is in Read-only Mode.

6.502.3.14 `virtual void cms::MapMessage::setByte (const std::string & name, unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Sets a Byte value with the specified name into the Map.

Parameters

name the name of the Byte

value the Byte value to set in the Map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2549) - if the **Message** (p. 2375) is in Read-only Mode.

6.502.3.15 `virtual void cms::MapMessage::setBytes (const std::string & name, const std::vector< unsigned char > & value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Sets a Bytes value with the specified name into the Map.

Parameters

name The name of the Bytes

value The Bytes value to set in the Map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2549) - if the **Message** (p. 2375) is in Read-only Mode.

6.502.3.16 `virtual void cms::MapMessage::setChar (const std::string & name, char value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Sets a Char value with the specified name into the Map.

Parameters

name the name of the Char

value the Char value to set in the Map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2549) - if the **Message** (p. 2375) is in Read-only Mode.

6.502.3.17 `virtual void cms::MapMessage::setDouble (const std::string & name, double value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Sets a Double value with the specified name into the Map.

Parameters

name The name of the Double

value The Double value to set in the Map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2549) - if the **Message** (p. 2375) is in Read-only Mode.

6.502.3.18 `virtual void cms::MapMessage::setFloat (const std::string & name, float value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Sets a Float value with the specified name into the Map.

Parameters

name The name of the Float

value The Float value to set in the Map

Exceptions

CMSException (p. 1074) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2549) - if the **Message** (p. 2375) is in Read-only Mode.

6.502.3.19 `virtual void cms::MapMessage::setInt (const std::string & name, int value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Sets a Int value with the specified name into the Map.

Parameters

name The name of the Int

value The Int value to set in the Map

Exceptions

CMSException (p. 1074) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2549) - if the **Message** (p. 2375) is in Read-only Mode.

6.502.3.20 `virtual void cms::MapMessage::setLong (const std::string & name, long long value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Sets a Long value with the specified name into the Map.

Parameters

name The name of the Long

value The Long value to set in the Map

Exceptions

CMSException (p. 1074) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2549) - if the **Message** (p. 2375) is in Read-only Mode.

6.502.3.21 `virtual void cms::MapMessage::setShort (const std::string & name, short value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Sets a Short value with the specified name into the Map.

Parameters

name The name of the Short

value The Short value to set in the Map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2549) - if the **Message** (p. 2375) is in Read-only Mode.

6.502.3.22 `virtual void cms::MapMessage::setString (const std::string & name, const std::string & value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Sets a String value with the specified name into the Map.

Parameters

name The name of the String

value The String value to set in the Map

Exceptions

CMSEException (p. 1074) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2549) - if the **Message** (p. 2375) is in Read-only Mode.

The documentation for this class was generated from the following file:

- `src/main/cms/MapMessage.h`

6.503 decaf::util::logging::MarkBlockLogger Class Reference

Defines a class that can be used to mark the entry and exit from scoped blocks.

```
#include <src/main/decaf/util/logging/MarkBlockLogger.h>
```

Public Member Functions

- **MarkBlockLogger** (**Logger** *logger, const std::string &blockName)
Constructor - Marks Block entry.
- virtual **~MarkBlockLogger** ()

6.503.1 Detailed Description

Defines a class that can be used to mark the entry and exit from scoped blocks. Create an instance of this class at the start of a scoped block, passing it the logger to use and the name of the block. The block entry and exit will be marked using the scope name, logger to the logger at the MARKBLOCK log level.

6.503.2 Constructor & Destructor Documentation

6.503.2.1 `decaf::util::logging::MarkBlockLogger::MarkBlockLogger (Logger * logger, const std::string & blockName) [inline]`

Constructor - Marks Block entry.

Parameters

- logger* **Logger** (p.2237) to use
- blockName* Block name

6.503.2.2 `virtual decaf::util::logging::MarkBlockLogger::~MarkBlockLogger () [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/MarkBlockLogger.h`

6.504 activemq::wireformat::MarshalAware Class Reference

```
#include <src/main/activemq/wireformat/MarshalAware.h>
```

Inheritance diagram for `activemq::wireformat::MarshalAware`:

Public Member Functions

- virtual `~MarshalAware ()`
- virtual bool `isMarshalAware () const =0`
Determine if the class implementing this interface is really wanting to be told about marshaling.
- virtual void `beforeMarshal (WireFormat *wireFormat)=0 throw (decaf::io::IOException)`
Called before marshaling is started to prepare the object to be marshaled.
- virtual void `afterMarshal (WireFormat *wireFormat)=0 throw (decaf::io::IOException)`
Called after marshaling is started to cleanup the object being marshaled.

- virtual void **beforeUnmarshal** (**WireFormat** *wireFormat)=0 throw (decaf::io::IOException)
Called before unmarshaling is started to prepare the object to be unmarshaled.
- virtual void **afterUnmarshal** (**WireFormat** *wireFormat)=0 throw (decaf::io::IOException)
Called after unmarshaling is started to cleanup the object being unmarshaled.
- virtual void **setMarshaledForm** (**WireFormat** *wireFormat, const std::vector< char > &data)=0
Called to set the data to this object that will contain the objects marshaled form.
- virtual std::vector< unsigned char > **getMarshaledForm** (**WireFormat** *wireFormat)=0
Called to get the data to this object that will contain the objects marshaled form.

6.504.1 Constructor & Destructor Documentation

- 6.504.1.1** virtual **activemq::wireformat::MarshalAware::~~MarshalAware** ()
 [inline, virtual]

6.504.2 Member Function Documentation

- 6.504.2.1** virtual void **activemq::wireformat::MarshalAware::afterMarshal** (**WireFormat** * *wireFormat*) throw (decaf::io::IOException) [pure virtual]

Called after marshaling is started to cleanup the object being marshaled.

Parameters

wireFormat - the wireformat object to control marshaling

- 6.504.2.2** virtual void **activemq::wireformat::MarshalAware::afterUnmarshal** (**WireFormat** * *wireFormat*) throw (decaf::io::IOException) [pure virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters

wireFormat - the wireformat object to control unmarshaling

- 6.504.2.3** virtual void **activemq::wireformat::MarshalAware::beforeMarshal** (**WireFormat** * *wireFormat*) throw (decaf::io::IOException) [pure virtual]

Called before marshaling is started to prepare the object to be marshaled.

Parameters

wireFormat - the wireformat object to control marshaling

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 319), and `activemq::commands::ActiveMQTextMessage` (p. 606).

6.504.2.4 `virtual void activemq::wireformat::MarshalAware::beforeUnmarshal (WireFormat * wireFormat) throw (decaf::io::IOException)` [pure virtual]

Called before unmarshaling is started to prepare the object to be unmarshaled.

Parameters

wireFormat - the wireformat object to control unmarshaling

6.504.2.5 `virtual std::vector<unsigned char> activemq::wireformat::MarshalAware::getMarshaledForm (WireFormat * wireFormat)` [pure virtual]

Called to get the data to this object that will contain the objects marshaled form.

Parameters

wireFormat - the wireformat object to control unmarshaling

Returns

buffer that holds the objects data.

6.504.2.6 `virtual bool activemq::wireformat::MarshalAware::isMarshalAware () const` [pure virtual]

Determine if the class implementing this interface is really wanting to be told about marshaling.

Normally if you didn't want to be marshal aware you just wouldn't implement this interface but since this is C++ and we don't have true interfaces we need a flat inheritance hierarchy, so we always implement this.

Returns

true if this class cares about marshaling.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 324), `activemq::commands::BaseDataStructure` (p. 767), `activemq::commands::Message` (p. 2368), and `activemq::commands::WireFormatInfo` (p. 3723).

6.504.2.7 `virtual void activemq::wireformat::MarshalAware::setMarshaledForm (WireFormat * wireFormat, const std::vector< char > & data)` [pure virtual]

Called to set the data to this object that will contain the objects marshaled form.

Parameters

- wireFormat* - the wireformat object to control unmarshaling
- data* - vector of object binary data

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/MarshalAware.h

6.505 activemq::wireformat::openwire::marshal::v6::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MarshallerFactory.h>
```

Public Member Functions

- virtual ~MarshallerFactory ()
- virtual void configure (OpenWireFormat *format)

6.505.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.505.2 Constructor & Destructor Documentation

6.505.2.1 virtual
activemq::wireformat::openwire::marshal::v6::MarshallerFactory::~MarshallerFactory
() [inline, virtual]

6.505.3 Member Function Documentation

6.505.3.1 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::MarshallerFactory::configure
(OpenWireFormat * *format*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/MarshallerFactory.h

6.506 activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h>
```

Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

6.506.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.506.2 Constructor & Destructor Documentation

6.506.2.1 virtual
`activemq::wireformat::openwire::marshal::v3::MarshallerFactory::~MarshallerFactory()` [inline, virtual]

6.506.3 Member Function Documentation

6.506.3.1 virtual void `activemq::wireformat::openwire::marshal::v3::MarshallerFactory::configure(OpenWireFormat * format)` [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h`

6.507 activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h>
```

Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

6.507.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.507.2 Constructor & Destructor Documentation

- 6.507.2.1** virtual
activemq::wireformat::openwire::marshal::v4::MarshallerFactory::~MarshallerFactory
() [inline, virtual]

6.507.3 Member Function Documentation

- 6.507.3.1** virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MarshallerFactory::configure
(OpenWireFormat * *format*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h

6.508 activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h>
```

Public Member Functions

- virtual ~MarshallerFactory ()
- virtual void configure (OpenWireFormat *format)

6.508.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.508.2 Constructor & Destructor Documentation

- 6.508.2.1** virtual
activemq::wireformat::openwire::marshal::v5::MarshallerFactory::~MarshallerFactory
() [inline, virtual]

6.508.3 Member Function Documentation

- 6.508.3.1** virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MarshallerFactory::configure
(OpenWireFormat * *format*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h

6.509 activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h>
```

Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

6.509.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.509.2 Constructor & Destructor Documentation

6.509.2.1 virtual
`activemq::wireformat::openwire::marshal::v1::MarshallerFactory::~~MarshallerFactory()` [inline, virtual]

6.509.3 Member Function Documentation

6.509.3.1 virtual void `activemq::wireformat::openwire::marshal::v1::MarshallerFactory::configure(OpenWireFormat * format)` [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h`

6.510 activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h>
```

Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

6.510.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.510.2 Constructor & Destructor Documentation

6.510.2.1 virtual
activemq::wireformat::openwire::marshal::v2::MarshallerFactory::~MarshallerFactory
 () [inline, virtual]

6.510.3 Member Function Documentation

6.510.3.1 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MarshallerFactory::configure
 (OpenWireFormat * *format*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MarshallerFactory.h**

6.511 activemq::util::MarshallingSupport Class Reference

```
#include <src/main/activemq/util/MarshallingSupport.h>
```

Public Member Functions

- **MarshallingSupport** ()
- virtual **~MarshallingSupport** ()

Static Public Member Functions

- static void **writeString** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw (decaf::io::IOException)
Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.
- static void **writeString16** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw (decaf::io::IOException)
Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.
- static void **writeString32** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw (decaf::io::IOException)
Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

- static std::string **readString16** (decaf::io::DataInputStream &dataIn) throw (decaf::io::IOException)

Reads an Openwire encoded string from the provided DataInputStream.

- static std::string **readString32** (decaf::io::DataInputStream &dataIn) throw (decaf::io::IOException)

Reads an Openwire encoded string from the provided DataInputStream.

- static std::string **asciiToModifiedUtf8** (const std::string &asciiString) throw (decaf::io::UTFDataFormatException)

Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string.

- static std::string **modifiedUtf8ToAscii** (const std::string modifiedUtf8String) throw (decaf::io::UTFDataFormatException)

Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255].

6.511.1 Constructor & Destructor Documentation

6.511.1.1 activemq::util::MarshallingSupport::MarshallingSupport ()

6.511.1.2 virtual activemq::util::MarshallingSupport::~~MarshallingSupport () [virtual]

6.511.2 Member Function Documentation

6.511.2.1 static std::string activemq::util::MarshallingSupport::asciiToModifiedUtf8 (const std::string & *asciiString*) throw (decaf::io::UTFDataFormatException) [static]

Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string.

This allows an ASCII string containing values greater than 127 as well as embedded NULLs to be sent to a Java client.

Parameters

asciiString The ASCII string to encode as Modified UTF-8

Returns

a string containing the Modified UTF-8 encoded form of the provided string.

Exceptions

UTFDataFormatException if the length of the encoded string would exceed the size of an signed integer.

6.511.2.2 `static std::string activemq::util::MarshallingSupport::modifiedUtf8ToAscii (const std::string modifiedUtf8String) throw (decaf::io::UTFDataFormatException) [static]`

Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255].

This will handle any string sent from a Java client which contains values within the [0..255] range or has embedded Nulls. Strings that have encoded values greater than 255 will cause an exception to be thrown.

Parameters

modifiedUtf8String The string to convert from Modified UTF-8 to ASCII.

Returns

the ASCII encoded version of the provided string.

Exceptions

UTFDataFormatException if the provided string contains invalid data or the character values encoded in the string exceed ASCII value 255.

6.511.2.3 `static std::string activemq::util::MarshallingSupport::readString16 (decaf::io::DataInputStream & dataIn) throw (decaf::io::IOException) [static]`

Reads an Openwire encoded string from the provided DataInputStream.

No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 16bits.

Parameters

dataIn The DataInputStream to read the String data from.

Returns

the String value.

Exceptions

IOException if an I/O error occurs while writing the string.

6.511.2.4 `static std::string activemq::util::MarshallingSupport::readString32 (decaf::io::DataInputStream & dataIn) throw (decaf::io::IOException) [static]`

Reads an Openwire encoded string from the provided DataInputStream.

No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 32bits.

Parameters

dataIn The DataInputStream to read the String data from.

Returns

the String value.

Exceptions

IOException if an I/O error occurs while writing the string.

```
6.511.2.5 static void activemq::util::MarshallingSupport::writeString (
    decaf::io::DataOutputStream & dataOut, const std::string & value )
    throw ( decaf::io::IOException ) [static]
```

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed.

Parameters

dataOut The DataOutputStream to write the String data to.

value Thre String value to write in Openwire form.

Exceptions

IOException if an I/O error occurs while writing the string.

```
6.511.2.6 static void activemq::util::MarshallingSupport::writeString16 (
    decaf::io::DataOutputStream & dataOut, const std::string & value )
    throw ( decaf::io::IOException ) [static]
```

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed. This method write out only the size as a short and the string data no Openwire Type tag is appended.

Parameters

dataOut The DataOutputStream to write the String data to.

value Thre String value to write in Openwire form.

Exceptions

IOException if an I/O error occurs while writing the string.

6.511.2.7 `static void activemq::util::MarshallingSupport::writeString32 (decaf::io::DataOutputStream & dataOut, const std::string & value) throw (decaf::io::IOException) [static]`

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed. This method write out only the size as a int and the string data no Openwire Type tag is appended.

Parameters

dataOut The DataOutputStream to write the String data to.

value Thre String value to write in Openwire form.

Exceptions

IOException if an I/O error occurs while writing the string.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/MarshallingSupport.h`

6.512 decaf::lang::Math Class Reference

The class `Math` (p. 2339) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

`#include <src/main/decaf/lang/Math.h>`

Public Member Functions

- `Math ()`
- `virtual ~Math ()`

Static Public Member Functions

- static int **abs** (int value)
Returns the absolute value of an int value.
- static long long **abs** (long long value)
Returns the absolute value of an long long value.
- static float **abs** (float value)
Returns the absolute value of a float value.
- static double **abs** (double value)
Returns the absolute value of a double value.
- static double **sqrt** (double value)
Returns the arc cosine of an angle, in the range of 0.0 through pi.

- static double **pow** (double base, double exp)
Returns the value of the first argument raised to the power of the second argument.
- static short **min** (short a, short b)
Returns the double value that is closest in value to the argument and is equal to a mathematical integer.
- static int **min** (int a, int b)
*Returns the smaller of two **int** values.*
- static unsigned int **min** (unsigned int a, unsigned int b)
*Returns the smaller of two **unsigned int** values.*
- static long long **min** (long long a, long long b)
*Returns the smaller of two **long long** values.*
- static float **min** (float a, float b)
Returns the smaller of two float values.
- static double **min** (double a, double b)
Returns the smaller of two double values.
- static short **max** (short a, short b)
*Returns the larger of two **short** values.*
- static int **max** (int a, int b)
*Returns the larger of two **int** values.*
- static long long **max** (long long a, long long b)
*Returns the larger of two **long long** values.*
- static float **max** (float a, float b)
Returns the greater of two float values.
- static double **max** (double a, double b)
Returns the greater of two double values.
- static double **ceil** (double value)
Returns the natural logarithm (base e) of a double value.
- static double **floor** (double value)
Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.
- static int **round** (float value)
Returns the closest int to the argument.
- static long long **round** (double value)
Returns the closest long long to the argument.

- static double **random** ()
Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.
- static float **signum** (float value)
Returns Euler's number e raised to the power of a double value.
- static double **signum** (double value)
Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.
- static double **toRadians** (double angdeg)
Returns the measure in radians of the supplied degree angle.
- static double **toDegrees** (double angrad)
Returns the measure in degrees of the supplied radian angle.

Static Public Attributes

- static const double **E**
- static const double **PI**

6.512.1 Detailed Description

The class `Math` (p. 2339) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

6.512.2 Constructor & Destructor Documentation

6.512.2.1 `decaf::lang::Math::Math ()` [inline]

6.512.2.2 `virtual decaf::lang::Math::~~Math ()` [inline, virtual]

6.512.3 Member Function Documentation

6.512.3.1 `static int decaf::lang::Math::abs (int value)` [inline, static]

Returns the absolute value of an int value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Parameters

value - the value to return the abs of

Returns

the value if positive, otherwise the negative of value

6.512.3.2 `static long long decaf::lang::Math::abs (long long value) [inline, static]`

Returns the absolute value of an long long value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Parameters

value - the value to return the abs of

Returns

the value if positive, otherwise the negative of value

6.512.3.3 `static double decaf::lang::Math::abs (double value) [static]`

Returns the absolute value of a double value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

- o If the argument is positive zero or negative zero, the result is positive zero.
- o If the argument is infinite, the result is positive infinity.
- o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: `Double::longBitsToDouble (p.1678)(0x7fffffffffffffffULL & Double::doubleToLongBits(value))`

Parameters

value - the value to return the abs of

Returns

the value if positive, otherwise the negative of value

6.512.3.4 `static float decaf::lang::Math::abs (float value) [static]`

Returns the absolute value of a float value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

- o If the argument is positive zero or negative zero, the result is positive zero.
- o If the argument is infinite, the result is positive infinity.
- o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: `Float::intBitsToFloat (p.1786)(0x7fffffff & Float::floatToIntBits(value))`

Parameters

value - the value to return the abs of

Returns

the value if positive, otherwise the negative of value

6.512.3.5 static double decaf::lang::Math::ceil (double *value*) [static]

Returns the natural logarithm (base e) of a double value.

Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity.

Parameters

value the value to compute the natural log of.

Returns

the natural log of value. Returns the base 10 logarithm of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity. o If the argument is equal to 10^n for integer n , then the result is n .

Parameters

value - the value to operate on

Returns

the long base 10 of value Returns the natural logarithm of the sum of the argument and 1. Note that for small values x , the result of $\log_{10}(x)$ is much closer to the true result of $\ln(1 + x)$ than the floating-point evaluation of $\log(1.0+x)$.

Special cases:

o If the argument is NaN or less than -1, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative one, then the result is negative infinity. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the value to operate on

Returns

the the value $\ln(x + 1)$, the natural log of $x + 1$ Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument. o If the argument value is less than zero but greater than -1.0, then the result is negative zero.

Note that the value of $\text{Math.ceil}(x)$ is exactly the value of $-\text{Math.floor}(-x)$.

Parameters

value - the value to find the ceiling of

Returns

the smallest (closest to negative infinity) floating-point value that is greater than or equal to the argument and is equal to a mathematical integer.

6.512.3.6 static double decaf::lang::Math::floor (double *value*) [static]

Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.

Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

Parameters

value - the value to find the floor of

Returns

the largest (closest to positive infinity) floating-point value that less than or equal to the argument and is equal to a mathematical integer.

6.512.3.7 static float decaf::lang::Math::max (float *a*, float *b*) [static]

Returns the greater of two float values.

That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

Parameters

a - an argument.

b - another argument.

Returns

the larger of *a* and *b*.

6.512.3.8 static double decaf::lang::Math::max (double *a*, double *b*) [static]

Returns the greater of two double values.

That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the larger of *a* and *b*.

6.512.3.9 `static short decaf::lang::Math::max (short a, short b) [inline, static]`

Returns the larger of two `short` values.

That is, the result the argument closer to the value of `Short::MAX_VALUE` (p. 3229). If the arguments have the same value, the result is that same value.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the larger of *a* and *b*.

6.512.3.10 `static int decaf::lang::Math::max (int a, int b) [inline, static]`

Returns the larger of two `int` values.

That is, the result the argument closer to the value of `Integer::MAX_VALUE` (p. 1956). If the arguments have the same value, the result is that same value.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the larger of *a* and *b*.

6.512.3.11 `static long long decaf::lang::Math::max (long long a, long long b) [inline, static]`

Returns the larger of two `long long` values.

That is, the result the argument closer to the value of `Long::MAX_VALUE` (p. 2281). If the arguments have the same value, the result is that same value.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the larger of **a** and **b**.

6.512.3.12 `static int decaf::lang::Math::min (int a, int b) [inline, static]`

Returns the smaller of two `int` values.

That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p.1956). If the arguments have the same value, the result is that same value.

Parameters

a - an argument.

b - another argument.

Returns

the smaller of **a** and **b**.

6.512.3.13 `static unsigned int decaf::lang::Math::min (unsigned int a, unsigned int b) [inline, static]`

Returns the smaller of two `unsigned int` values.

That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p.1956). If the arguments have the same value, the result is that same value.

Parameters

a - an argument.

b - another argument.

Returns

the smaller of **a** and **b**.

6.512.3.14 `static long long decaf::lang::Math::min (long long a, long long b) [inline, static]`

Returns the smaller of two `long long` values.

That is, the result the argument closer to the value of `Long::MIN_VALUE` (p.2281). If the arguments have the same value, the result is that same value.

Parameters

a - an argument.

b - another argument.

Returns

the smaller of **a** and **b**.

6.512.3.15 static float decaf::lang::Math::min (float *a*, float *b*) [static]

Returns the smaller of two float values.

That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the smaller of *a* and *b*.

6.512.3.16 static double decaf::lang::Math::min (double *a*, double *b*) [static]

Returns the smaller of two double values.

That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the smaller of *a* and *b*.

6.512.3.17 static short decaf::lang::Math::min (short *a*, short *b*) [inline, static]

Returns the double value that is closest in value to the argument and is equal to a mathematical integer.

If two double values that are mathematical integers are equally close, the result is the integer value that is even. Special cases:

- o If the argument value is already equal to a mathematical integer, then the result is the same as the argument.
- o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

Parameters

- value* - the value to round to the nearest integer

Returns

the rounded value Returns the smaller of two short values. That is, the result is the argument closer to the value of `decaf.lang.Short::MIN_VALUE` (p. 3229). If the arguments have the same value, the result is that same value.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the smaller of *a* and *b*.

6.512.3.18 `static double decaf::lang::Math::pow (double base, double exp)`
[static]

Returns the value of the first argument raised to the power of the second argument.

Special cases:

- o If the second argument is positive or negative zero, then the result is 1.0.
- o If the second argument is 1.0, then the result is the same as the first argument.
- o If the second argument is NaN, then the result is NaN.
- o If the first argument is NaN and the second argument is nonzero, then the result is NaN.

Parameters

- base* - the base
- exp* - the exponent

Returns

the base raised to the power of *exp*.

6.512.3.19 `static double decaf::lang::Math::random ()` [static]

Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.

The remainder value is mathematically equal to $f1 - f2 \times n$, where *n* is the mathematical integer closest to the exact mathematical value of the quotient $f1/f2$, and if two mathematical integers are equally close to $f1/f2$, then *n* is the integer that is even. If the remainder is zero, its sign is the same as the sign of the first argument. Special cases:

- o If either argument is NaN, or the first argument is infinite, or the second argument is positive zero or negative zero, then the result is NaN.
- o If the first argument is finite and the second argument is infinite, then the result is the same as the first argument.

Parameters

- f1* - the dividend.
- f2* - the divisor

Returns

the IEEE remainder of value Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range.

When this method is first called, it creates a single new pseudorandom-number generator; This new pseudorandom-number generator is used thereafter for all calls to this method and is used nowhere else.

This method is properly synchronized to allow correct use by more than one thread. However, if many threads need to generate pseudorandom numbers at a great rate, it may reduce contention for each thread to have its own pseudorandom-number generator.

Returns

a pseudorandom double greater than or equal to 0.0 and less than 1.0.

6.512.3.20 static long long decaf::lang::Math::round (double *value*) [static]

Returns the closest long long to the argument.

The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type long long. In other words, the result is equal to the value of the expression: (long long)**Math.floor** (p. 2344)(a + 0.5d)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Long::MIN_VALUE** (p. 2281), the result is equal to the value of **Long::MIN_VALUE** (p. 2281). o If the argument is positive infinity or any value greater than or equal to the value of **Long::MAX_VALUE** (p. 2281), the result is equal to the value of **Long::MAX_VALUE** (p. 2281).

Parameters

value - the value to round

Returns

the value of the argument rounded to the nearest integral value.

6.512.3.21 static int decaf::lang::Math::round (float *value*) [static]

Returns the closest int to the argument.

The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type int. In other words, the result is equal to the value of the expression: (int)**Math.floor** (p. 2344)(a + 0.5f)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Integer::MIN_VALUE** (p. 1956), the result is equal to the value of **Integer::MIN_VALUE** (p. 1956). o If the argument is positive infinity or any value greater than or equal to the value of **Integer::MAX_VALUE** (p. 1956), the result is equal to the value of **Integer::MAX_VALUE** (p. 1956).

Parameters

value - the value to round

Returns

the value of the argument rounded to the nearest integral value.

6.512.3.22 static double decaf::lang::Math::signum (double *value*) [static]

Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.

Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Parameters

value - the floating-point value whose signum is to be returned

Returns

the signum function of the argument

6.512.3.23 static float decaf::lang::Math::signum (float *value*) [static]

Returns Euler's number e raised to the power of a double value.

Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is positive zero.

Parameters

value - the exponent to raise e to

Returns

the value e^x , where e is the base of the natural logarithms. Returns $e^x - 1$. Note that for values of x near 0, the exact sum of $\expm1(x) + 1$ is much closer to the true result of e^x than $\exp(x)$. Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is -1.0. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the value to raise $e^x - 1$

Returns

the value $e^x - 1$. Returns $\sqrt{x^2 + y^2}$ without intermediate overflow or underflow. Special cases:

If either argument is infinite, then the result is positive infinity. If either argument is NaN and neither argument is infinite, then the result is NaN.

Parameters

x - an argument

y - another argument

Returns

the $\sqrt{x^2 + y^2}$ without intermediate overflow or underflow Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero. Special Cases:

- o If the argument is NaN, then the result is NaN.
- o If the argument is positive zero or negative zero, then the result is the same as the argument.

Parameters

value - the floating-point value whose signum is to be returned

Returns

the signum function of the argument

6.512.3.24 static double decaf::lang::Math::sqrt (double *value*) [static]

Returns the arc cosine of an angle, in the range of 0.0 through pi.

Special case:

- o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.

Parameters

value - the value to return the arc cosine of.

Returns

arc cosine of value in radians. Returns the arc sine of an angle, in the range of -pi/2 through pi/2. Special cases:

- o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.
- o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the value to return the arc cosine of.

Returns

arc cosine of value in radians. Returns the arc tangent of an angle, in the range of -pi/2 through pi/2. Special cases:

- o If the argument is NaN, then the result is NaN.
- o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the value to return the arc cosine of.

Returns

arc tangent of value in radians. Converts rectangular coordinates (x, y) to polar (r, theta). This method computes the phase theta by computing an arc tangent of y/x in the range of -pi to pi. Special cases:

o If either argument is NaN, then the result is NaN. o If the first argument is positive zero and the second argument is positive, or the first argument is positive and finite and the second argument is positive infinity, then the result is positive zero. o If the first argument is negative zero and the second argument is positive, or the first argument is negative and finite and the second argument is positive infinity, then the result is negative zero. o If the first argument is positive zero and the second argument is negative, or the first argument is positive and finite and the second argument is negative infinity, then the result is the double value closest to pi. o If the first argument is negative zero and the second argument is negative, or the first argument is negative and finite and the second argument is negative infinity, then the result is the double value closest to -pi. o If the first argument is positive and the second argument is positive zero or negative zero, or the first argument is positive infinity and the second argument is finite, then the result is the double value closest to pi/2. o If the first argument is negative and the second argument is positive zero or negative zero, or the first argument is negative infinity and the second argument is finite, then the result is the double value closest to -pi/2. o If both arguments are positive infinity, then the result is the double value closest to pi/4. o If the first argument is positive infinity and the second argument is negative infinity, then the result is the double value closest to 3*pi/4. o If the first argument is negative infinity and the second argument is positive infinity, then the result is the double value closest to -pi/4. o If both arguments are negative infinity, then the result is the double value closest to -3*pi/4.

Parameters

y - the ordinate coordinate
x - the abscissa coordinate

Returns

the theta component of the point (r, theta) in polar coordinates that corresponds to the point (x, y) in Cartesian coordinates. Returns the cube root of a double value. For positive finite x, $\text{cbrt}(-x) == -\text{cbrt}(x)$; that is, the cube root of a negative value is the negative of the cube root of that value's magnitude. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the double to compute the cube root of

Returns

the cube root of value Returns the trigonometric cosine of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN.

Parameters

value - an value in radians

Returns

the cosine of the argument. Returns the hyperbolic cosine of a double value. The hyperbolic cosine of x is defined to be $(e^x + e^{-x})/2$ where e is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is positive infinity. o If the argument is zero, then the result is 1.0.

Parameters

value - the number whose hyperbolic cosine is to be found

Returns

the hyperbolic cosine of value Returns the trigonometric sine of an angle. Special case:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the number whose sin is to be found

Returns

the sine of value Returns the hyperbolic sine of a double value. The hyperbolic sine of x is defined to be $(e^x - e^{-x})/2$ where e is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the number whose hyperbolic sin is to be found

Returns

the hyperbolic sine of value Returns the trigonometric tangent of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the number whose tangent is to be found

Returns

the tangent of value Returns the hyperbolic tangent of a double value. The hyperbolic tangent of x is defined to be $(e^x - e^{-x})/(e^x + e^{-x})$, in other words, $\sinh(x)/\cosh(x)$. Note that the absolute value of the exact tanh is always less than 1. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument. o If the argument is positive infinity, then the result is +1.0. o If the argument is negative infinity, then the result is -1.0.

Parameters

value - the number whose hyperbolic tangent is to be found

Returns

the hyperbolic cosine of value Returns the correctly rounded positive square root of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the double value closest to the true mathematical square root of the argument value.

Parameters

value - the value to find the square root of
the square root of the argument.

6.512.3.25 `static double decaf::lang::Math::toDegrees (double angrad)` [inline, static]

Returns the measure in degrees of the supplied radian angle.

Parameters

angrad - an angle in radians

Returns

the degree measure of the angle.

6.512.3.26 `static double decaf::lang::Math::toRadians (double angdeg)` [inline, static]

Returns the measure in radians of the supplied degree angle.

Parameters

angdeg - an angle in degrees

Returns

the radian measure of the angle.

6.512.4 Field Documentation

6.512.4.1 `const double decaf::lang::Math::E` [static]

6.512.4.2 `const double decaf::lang::Math::PI` [static]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Math.h`

6.513 activemq::util::MemoryUsage Class Reference

```
#include <src/main/activemq/util/MemoryUsage.h>
```

Inheritance diagram for `activemq::util::MemoryUsage`:

Public Member Functions

- **MemoryUsage** ()
Default Constructor.
- **MemoryUsage** (unsigned long long limit)
*Creates an instance of an **Usage** (p. 3701) monitor with a set limit.*
- virtual **~MemoryUsage** ()
- virtual void **waitForSpace** ()
*Waits forever for more space to be returned to this **Usage** (p. 3701) Manager.*
- virtual void **waitForSpace** (unsigned int timeout)
*Waits for more space to be returned to this **Usage** (p. 3701) Manager, times out when the given time span in milliseconds elapses.*
- virtual void **enqueueUsage** (unsigned long long value)
Tries to increase the usage by value amount but blocks if this object is currently full.
- virtual void **increaseUsage** (unsigned long long value)
Increases the usage by the value amount.
- virtual void **decreaseUsage** (unsigned long long value)
Decreases the usage by the value amount.
- virtual bool **isFull** () const
*Returns true if this **Usage** (p. 3701) instance is full, i.e.*
- unsigned long long **getUsage** () const
Gets the current usage amount.
- void **setUsage** (unsigned long long usage)
Sets the current usage amount.
- unsigned long long **getLimit** () const
Gets the current limit amount.
- void **setLimit** (unsigned long long limit)
Sets the current limit amount.

6.513.1 Constructor & Destructor Documentation

6.513.1.1 activemq::util::MemoryUsage::MemoryUsage ()

Default Constructor.

6.513.1.2 `activemq::util::MemoryUsage::MemoryUsage (unsigned long long limit)`

Creates an instance of an **Usage** (p. 3701) monitor with a set limit.

Parameters

limit - amount of memory this manager allows.

6.513.1.3 `virtual activemq::util::MemoryUsage::~~MemoryUsage () [virtual]`

6.513.2 Member Function Documentation

6.513.2.1 `virtual void activemq::util::MemoryUsage::decreaseUsage (unsigned long long value) [virtual]`

Decreases the usage by the value amount.

Parameters

value Amount of space to return to the pool

Implements **activemq::util::Usage** (p. 3702).

6.513.2.2 `virtual void activemq::util::MemoryUsage::enqueueUsage (unsigned long long value) [inline, virtual]`

Tries to increase the usage by value amount but blocks if this object is currently full.

Parameters

value Amount of usage in bytes to add.

Implements **activemq::util::Usage** (p. 3702).

6.513.2.3 `unsigned long long activemq::util::MemoryUsage::getLimit () const [inline]`

Gets the current limit amount.

Returns

the amount that can be used before full.

6.513.2.4 `unsigned long long activemq::util::MemoryUsage::getUsage () const [inline]`

Gets the current usage amount.

Returns

the amount of bytes currently used.

6.513.2.5 `virtual void activemq::util::MemoryUsage::increaseUsage (unsigned long long value) [virtual]`

Increases the usage by the value amount.

Parameters

value Amount of usage to add.

Implements `activemq::util::Usage` (p. 3702).

6.513.2.6 `virtual bool activemq::util::MemoryUsage::isFull () const [virtual]`

Returns true if this `Usage` (p. 3701) instance is full, i.e.

`Usage` (p. 3701) $\geq 100\%$

Implements `activemq::util::Usage` (p. 3703).

6.513.2.7 `void activemq::util::MemoryUsage::setLimit (unsigned long long limit) [inline]`

Sets the current limit amount.

Parameters

limit - The amount that can be used before full.

6.513.2.8 `void activemq::util::MemoryUsage::setUsage (unsigned long long usage) [inline]`

Sets the current usage amount.

Parameters

usage - The amount to tag as used.

6.513.2.9 `virtual void activemq::util::MemoryUsage::waitForSpace () [virtual]`

Waits forever for more space to be returned to this `Usage` (p. 3701) Manager.

Implements `activemq::util::Usage` (p. 3703).

6.513.2.10 `virtual void activemq::util::MemoryUsage::waitForSpace (unsigned int timeout) [virtual]`

Waits for more space to be returned to this `Usage` (p. 3701) Manager, times out when the given time span in milliseconds elapses.

Parameters

timeout The time to wait for more space.

Implements **activemq::util::Usage** (p. 3703).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/MemoryUsage.h`

6.514 **activemq::commands::Message** Class Reference

```
#include <src/main/activemq/commands/Message.h>
```

Inheritance diagram for **activemq::commands::Message**:

Public Member Functions

- **Message** ()
- virtual **~Message** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **Message * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)
Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.
- virtual void **afterUnmarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)
Called after unmarshaling is started to cleanup the object being unmarshaled.
- virtual bool **isMarshalAware** () const
Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.
- virtual void **setAckHandler** (const **Pointer**< **core::ActiveMQAckHandler** > &handler)

*Sets the Acknowledgment Handler that this **Message** (p. 2358) will use when the Acknowledge method is called.*

- virtual **Pointer**< **core::ActiveMQAckHandler** > **getAckHandler** () const
*Gets the Acknowledgment Handler that this **Message** (p. 2358) will use when the Acknowledge method is called.*
- void **setConnection** (**core::ActiveMQConnection** ***connection**)
*Sets the ActiveMQConnection instance that this **Command** (p. 1107) was created from when the session create methods are called to create a **Message** (p. 2358).*
- **core::ActiveMQConnection** * **getConnection** () const
*Gets the ActiveMQConnection instance that this **Command** (p. 1107) was created from when the session create methods are called to create a **Message** (p. 2358).*
- virtual unsigned int **getSize** () const
Returns the Size of this message in Bytes.
- virtual bool **isExpired** () const
Returns if this message has expired, meaning that its Expiration time has elapsed.
- virtual void **onSend** ()
*Allows derived **Message** (p. 2358) classes to perform tasks before a message is sent.*
- **util::PrimitiveMap** & **getMessageProperties** ()
Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.
- const **util::PrimitiveMap** & **getMessageProperties** () const
- bool **isReadOnlyProperties** () const
*Returns if the **Message** (p. 2358) Properties Are Read Only.*
- void **setReadOnlyProperties** (bool value)
*Set the Read Only State of the **Message** (p. 2358) Properties.*
- bool **isReadOnlyBody** () const
*Returns if the **Message** (p. 2358) Body is Read Only.*
- void **setReadOnlyBody** (bool value)
*Set the Read Only State of the **Message** (p. 2358) Content.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &**producerId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &**transactionId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** () const

- virtual **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** ()
- virtual void **setOriginalDestination** (const **Pointer**< **ActiveMQDestination** > &**originalDestination**)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)
- virtual const **Pointer**< **TransactionId** > & **getOriginalTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getOriginalTransactionId** ()
- virtual void **setOriginalTransactionId** (const **Pointer**< **TransactionId** > &**originalTransactionId**)
- virtual const std::string & **getGroupID** () const
- virtual std::string & **getGroupID** ()
- virtual void **setGroupID** (const std::string &**groupID**)
- virtual int **getGroupSequence** () const
- virtual void **setGroupSequence** (int **groupSequence**)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &**correlationId**)
- virtual bool **isPersistent** () const
- virtual void **setPersistent** (bool **persistent**)
- virtual long long **getExpiration** () const
- virtual void **setExpiration** (long long **expiration**)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char **priority**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getReplyTo** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getReplyTo** ()
- virtual void **setReplyTo** (const **Pointer**< **ActiveMQDestination** > &**replyTo**)
- virtual long long **getTimestamp** () const
- virtual void **setTimestamp** (long long **timestamp**)
- virtual const std::string & **getType** () const
- virtual std::string & **getType** ()
- virtual void **setType** (const std::string &**type**)
- virtual const std::vector< unsigned char > & **getContent** () const
- virtual std::vector< unsigned char > & **getContent** ()
- virtual void **setContent** (const std::vector< unsigned char > &**content**)
- virtual const std::vector< unsigned char > & **getMarshaledProperties** () const
- virtual std::vector< unsigned char > & **getMarshaledProperties** ()
- virtual void **setMarshaledProperties** (const std::vector< unsigned char > &**marshalledProperties**)
- virtual const **Pointer**< **DataStructure** > & **getDataStructure** () const
- virtual **Pointer**< **DataStructure** > & **getDataStructure** ()
- virtual void **setDataStructure** (const **Pointer**< **DataStructure** > &**dataStructure**)
- virtual const **Pointer**< **ConsumerId** > & **getTargetConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getTargetConsumerId** ()
- virtual void **setTargetConsumerId** (const **Pointer**< **ConsumerId** > &**targetConsumerId**)
- virtual bool **isCompressed** () const
- virtual void **setCompressed** (bool **compressed**)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int **redeliveryCounter**)

- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**brokerPath**)
- virtual long long **getArrival** () const
- virtual void **setArrival** (long long **arrival**)
- virtual const std::string & **getUserID** () const
- virtual std::string & **getUserID** ()
- virtual void **setUserID** (const std::string &**userID**)
- virtual bool **isRecievedByDFBridge** () const
- virtual void **setRecievedByDFBridge** (bool **recievedByDFBridge**)
- virtual bool **isDroppable** () const
- virtual void **setDroppable** (bool **droppable**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getCluster** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getCluster** ()
- virtual void **setCluster** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**cluster**)
- virtual long long **getBrokerInTime** () const
- virtual void **setBrokerInTime** (long long **brokerInTime**)
- virtual long long **getBrokerOutTime** () const
- virtual void **setBrokerOutTime** (long long **brokerOutTime**)
- virtual bool **isMessage** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** ***visitor**) throw (**exceptions::ActiveMQException**)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _MESSAGE** = 0

Protected Attributes

- **core::ActiveMQConnection** * **connection**
- **Pointer**< **ProducerId** > **producerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **TransactionId** > **transactionId**
- **Pointer**< **ActiveMQDestination** > **originalDestination**
- **Pointer**< **MessageId** > **messageId**
- **Pointer**< **TransactionId** > **originalTransactionId**
- std::string **groupId**
- int **groupSequence**
- std::string **correlationId**
- bool **persistent**
- long long **expiration**
- unsigned char **priority**
- **Pointer**< **ActiveMQDestination** > **replyTo**
- long long **timestamp**

- `std::string` **type**
- `std::vector< unsigned char >` **content**
- `std::vector< unsigned char >` **marshalledProperties**
- `Pointer< DataStructure >` **dataStructure**
- `Pointer< ConsumerId >` **targetConsumerId**
- `bool` **compressed**
- `int` **redeliveryCounter**
- `std::vector< decaf::lang::Pointer< BrokerId > >` **brokerPath**
- `long long` **arrival**
- `std::string` **userID**
- `bool` **recievedByDFBridge**
- `bool` **droppable**
- `std::vector< decaf::lang::Pointer< BrokerId > >` **cluster**
- `long long` **brokerInTime**
- `long long` **brokerOutTime**

Static Protected Attributes

- `static const unsigned int` **DEFAULT_MESSAGE_SIZE** = 1024

6.514.1 Constructor & Destructor Documentation

6.514.1.1 `activemq::commands::Message::Message ()`

6.514.1.2 `virtual activemq::commands::Message::~Message ()` [virtual]

6.514.2 Member Function Documentation

6.514.2.1 `virtual void activemq::commands::Message::afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)` [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters

wireFormat - the wireformat object to control unmarshaling

Reimplemented from `activemq::commands::BaseDataStructure` (p. 765).

6.514.2.2 `virtual void activemq::commands::Message::beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)` [virtual]

Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.

Parameters

wireFormat - the wireformat controller

Reimplemented from `activemq::commands::BaseDataStructure` (p. 765).

6.514.2.3 virtual Message* activemq::commands::Message::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 168), **activemq::commands::ActiveMQBytesMessage** (p. 198), **activemq::commands::ActiveMQMapMessage** (p. 320), **activemq::commands::ActiveMQMessage** (p. 354), **activemq::commands::ActiveMQObjectMessage** (p. 397), **activemq::commands::ActiveMQStreamMessage** (p. 489), and **activemq::commands::ActiveMQTextMessage** (p. 607).

6.514.2.4 virtual void activemq::commands::Message::copyDataStructure (const DataStructure * src) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 168), **activemq::commands::ActiveMQBytesMessage** (p. 198), **activemq::commands::ActiveMQMapMessage** (p. 320), **activemq::commands::ActiveMQMessage** (p. 354), **activemq::commands::ActiveMQObjectMessage** (p. 398), **activemq::commands::ActiveMQStreamMessage** (p. 489), and **activemq::commands::ActiveMQTextMessage** (p. 607).

6.514.2.5 virtual bool activemq::commands::Message::equals (const DataStructure * value) const [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 168), **activemq::commands::ActiveMQBytesMessage** (p. 199), **activemq::commands::ActiveMQMapMessage** (p. 320), **activemq::commands::ActiveMQMessage** (p. 354), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 383), and **activemq::commands::ActiveMQObjectMessage**

(p. 398), `activemq::commands::ActiveMQStreamMessage` (p. 490), `activemq::commands::ActiveMQTextMessage` (p. 607), `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 383), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 383), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 383), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 383), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 383), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 383).

6.514.2.6 `virtual Pointer<core::ActiveMQAckHandler>
activemq::commands::Message::getAckHandler () const [inline,
virtual]`

Gets the Acknowledgment Handler that this **Message** (p. 2358) will use when the Acknowledge method is called.

Returns

handler `ActiveMQAckHandler` to call or `NULL` if not set

6.514.2.7 `virtual long long activemq::commands::Message::getArrival () const
[virtual]`

6.514.2.8 `virtual long long activemq::commands::Message::getBrokerInTime ()
const [virtual]`

6.514.2.9 `virtual long long activemq::commands::Message::getBrokerOutTime ()
const [virtual]`

6.514.2.10 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >&
activemq::commands::Message::getBrokerPath () const [virtual]`

6.514.2.11 `virtual std::vector< decaf::lang::Pointer<BrokerId> >&
activemq::commands::Message::getBrokerPath () [virtual]`

6.514.2.12 `virtual std::vector< decaf::lang::Pointer<BrokerId> >&
activemq::commands::Message::getCluster () [virtual]`

6.514.2.13 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >&
activemq::commands::Message::getCluster () const [virtual]`

6.514.2.14 `core::ActiveMQConnection* ac-
tivemq::commands::Message::getConnection () const
[inline]`

Gets the `ActiveMQConnection` instance that this **Command** (p. 1107) was created from when the session create methods are called to create a **Message** (p. 2358).

Returns

the `ActiveMQConnection` parent for this **Message** (p. 2358) or `NULL` if not set.

- 6.514.2.15 `virtual const std::vector<unsigned char>& activemq::commands::Message::getContent () const`
[virtual]
- 6.514.2.16 `virtual std::vector<unsigned char>& activemq::commands::Message::getContent ()`
[virtual]
- 6.514.2.17 `virtual const std::string& activemq::commands::Message::getCorrelationId () const` [virtual]
- 6.514.2.18 `virtual std::string& activemq::commands::Message::getCorrelationId ()` [virtual]
- 6.514.2.19 `virtual Pointer<DataStructure>& activemq::commands::Message::getDataStructure ()`
[virtual]
- 6.514.2.20 `virtual const Pointer<DataStructure>& activemq::commands::Message::getDataStructure () const`
[virtual]
- 6.514.2.21 `virtual unsigned char activemq::commands::Message::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 168), **activemq::commands::ActiveMQBytesMessage** (p. 200), **activemq::commands::ActiveMQMapMessage** (p. 322), **activemq::commands::ActiveMQMessage** (p. 354), **activemq::commands::ActiveMQObjectMessage** (p. 398), **activemq::commands::ActiveMQStreamMessage** (p. 490), and **activemq::commands::ActiveMQTextMessage** (p. 607).

- 6.514.2.22 `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination () const [virtual]`
- 6.514.2.23 `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination () [virtual]`
- 6.514.2.24 `virtual long long activemq::commands::Message::getExpiration () const [virtual]`
- 6.514.2.25 `virtual const std::string& activemq::commands::Message::getGroupID () const [virtual]`
- 6.514.2.26 `virtual std::string& activemq::commands::Message::getGroupID () [virtual]`
- 6.514.2.27 `virtual int activemq::commands::Message::getGroupSequence () const [virtual]`
- 6.514.2.28 `virtual const std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties () const [virtual]`
- 6.514.2.29 `virtual std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties () [virtual]`
- 6.514.2.30 `virtual const Pointer<MessageId>& activemq::commands::Message::getMessageId () const [virtual]`
- 6.514.2.31 `virtual Pointer<MessageId>& activemq::commands::Message::getMessageId () [virtual]`
- 6.514.2.32 `util::PrimitiveMap& activemq::commands::Message::getMessageProperties () [inline]`

Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.

Returns

a reference to the Primitive Map that holds message properties.

- 6.514.2.33 `const util::PrimitiveMap& activemq::commands::Message::getMessageProperties ()`
`const [inline]`
- 6.514.2.34 `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination ()`
`[virtual]`
- 6.514.2.35 `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination () const`
`[virtual]`
- 6.514.2.36 `virtual const Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId () const`
`[virtual]`
- 6.514.2.37 `virtual Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId ()`
`[virtual]`
- 6.514.2.38 `virtual unsigned char activemq::commands::Message::getPriority ()`
`const [virtual]`
- 6.514.2.39 `virtual const Pointer<ProducerId>& activemq::commands::Message::getProducerId () const`
`[virtual]`
- 6.514.2.40 `virtual Pointer<ProducerId>& activemq::commands::Message::getProducerId ()`
`[virtual]`
- 6.514.2.41 `virtual int activemq::commands::Message::getRedeliveryCounter ()`
`const [virtual]`
- 6.514.2.42 `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo () const` `[virtual]`
- 6.514.2.43 `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo ()` `[virtual]`
- 6.514.2.44 `virtual unsigned int activemq::commands::Message::getSize () const`
`[virtual]`

Returns the Size of this message in Bytes.

Returns

number of bytes this message equates to.

Reimplemented in `activemq::commands::ActiveMQTextMessage` (p. 608).

- 6.514.2.45 `virtual const Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId () const [virtual]`
- 6.514.2.46 `virtual Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId () [virtual]`
- 6.514.2.47 `virtual long long activemq::commands::Message::getTimestamp () const [virtual]`
- 6.514.2.48 `virtual const Pointer<TransactionId>& activemq::commands::Message::getTransactionId () const [virtual]`
- 6.514.2.49 `virtual Pointer<TransactionId>& activemq::commands::Message::getTransactionId () [virtual]`
- 6.514.2.50 `virtual const std::string& activemq::commands::Message::getType () const [virtual]`
- 6.514.2.51 `virtual std::string& activemq::commands::Message::getType () [virtual]`
- 6.514.2.52 `virtual const std::string& activemq::commands::Message::getUserID () const [virtual]`
- 6.514.2.53 `virtual std::string& activemq::commands::Message::getUserID () [virtual]`
- 6.514.2.54 `virtual bool activemq::commands::Message::isCompressed () const [virtual]`
- 6.514.2.55 `virtual bool activemq::commands::Message::isDroppable () const [virtual]`
- 6.514.2.56 `virtual bool activemq::commands::Message::isExpired () const [virtual]`

Returns if this message has expired, meaning that its Expiration time has elapsed.

Returns

true if message is expired.

- 6.514.2.57 `virtual bool activemq::commands::Message::isMarshalAware () const [inline, virtual]`

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

Returns

boolean indicating desire to be in marshaling stages

Reimplemented from `activemq::commands::BaseDataStructure` (p. 767).

Reimplemented in `activemq::commands::ActiveMQMapMessage` (p. 324).

6.514.2.58 `virtual bool activemq::commands::Message::isMessage () const`
[inline, virtual]

Returns

an answer of true to the `isMessage()` (p. 2369) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 698).

6.514.2.59 `virtual bool activemq::commands::Message::isPersistent () const`
[virtual]

6.514.2.60 `bool activemq::commands::Message::isReadOnlyBody () const`
[inline]

Returns if the `Message` (p. 2358) Body is Read Only.

Returns

true if `Message` (p. 2358) Content is Read Only.

6.514.2.61 `bool activemq::commands::Message::isReadOnlyProperties () const`
[inline]

Returns if the `Message` (p. 2358) Properties Are Read Only.

Returns

true if `Message` (p. 2358) Properties are Read Only.

6.514.2.62 `virtual bool activemq::commands::Message::isRecievedByDFBridge () const` [virtual]

6.514.2.63 `virtual void activemq::commands::Message::onSend ()` [inline, virtual]

Allows derived `Message` (p. 2358) classes to perform tasks before a message is sent.

Reimplemented in `activemq::commands::ActiveMQBytesMessage` (p. 200),
`activemq::commands::ActiveMQMessageTemplate< T >` (p. 390),
`activemq::commands::ActiveMQStreamMessage` (p. 490),
`activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 390),
`activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 390),
`activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 390),
`activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 390),
`activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 390), and
`activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 390).

6.514.2.64 `virtual void activemq::commands::Message::setAckHandler (const
Pointer< core::ActiveMQAckHandler > & handler) [inline,
virtual]`

Sets the Acknowledgment Handler that this **Message** (p. 2358) will use when the Acknowledge method is called.

Parameters

handler ActiveMQAckHandler to call

6.514.2.65 `virtual void activemq::commands::Message::setArrival (long long
arrival) [virtual]`

6.514.2.66 `virtual void activemq::commands::Message::setBrokerInTime (long
long brokerInTime) [virtual]`

6.514.2.67 `virtual void activemq::commands::Message::setBrokerOutTime (long
long brokerOutTime) [virtual]`

6.514.2.68 `virtual void activemq::commands::Message::setBrokerPath (const
std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)
[virtual]`

6.514.2.69 `virtual void activemq::commands::Message::setCluster (const
std::vector< decaf::lang::Pointer< BrokerId > > & cluster) [virtual]`

6.514.2.70 `virtual void activemq::commands::Message::setCompressed (bool
compressed) [virtual]`

6.514.2.71 `void activemq::commands::Message::setConnection (const
core::ActiveMQConnection * connection) [inline]`

Sets the ActiveMQConnection instance that this **Command** (p. 1107) was created from when the session create methods are called to create a **Message** (p. 2358).

Parameters

handler ActiveMQConnection parent for this message

- 6.514.2.72 virtual void activemq::commands::Message::setContent (const std::vector< unsigned char > & *content*) [virtual]
- 6.514.2.73 virtual void activemq::commands::Message::setCorrelationId (const std::string & *correlationId*) [virtual]
- 6.514.2.74 virtual void activemq::commands::Message::setDataStructure (const Pointer< DataStructure > & *dataStructure*) [virtual]
- 6.514.2.75 virtual void activemq::commands::Message::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.514.2.76 virtual void activemq::commands::Message::setDroppable (bool *droppable*) [virtual]
- 6.514.2.77 virtual void activemq::commands::Message::setExpiration (long long *expiration*) [virtual]
- 6.514.2.78 virtual void activemq::commands::Message::setGroupID (const std::string & *groupID*) [virtual]
- 6.514.2.79 virtual void activemq::commands::Message::setGroupSequence (int *groupSequence*) [virtual]
- 6.514.2.80 virtual void activemq::commands::Message::setMarshaledProperties (const std::vector< unsigned char > & *marshalledProperties*) [virtual]
- 6.514.2.81 virtual void activemq::commands::Message::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]
- 6.514.2.82 virtual void activemq::commands::Message::setOriginalDestination (const Pointer< ActiveMQDestination > & *originalDestination*) [virtual]
- 6.514.2.83 virtual void activemq::commands::Message::setOriginalTransactionId (const Pointer< TransactionId > & *originalTransactionId*) [virtual]
- 6.514.2.84 virtual void activemq::commands::Message::setPersistent (bool *persistent*) [virtual]
- 6.514.2.85 virtual void activemq::commands::Message::setPriority (unsigned char *priority*) [virtual]
- 6.514.2.86 virtual void activemq::commands::Message::setProducerId (const Pointer< ProducerId > & *producerId*) [virtual]
- 6.514.2.87 void activemq::commands::Message::setReadOnlyBody (bool *value*) [inline]

Set the Read Only State of the **Message** (p. 2358) Content.

Parameters

value - true if Content should be read only.

6.514.2.88 `void activemq::commands::Message::setReadOnlyProperties (bool value) [inline]`

Set the Read Only State of the **Message** (p. 2358) Properties.

Parameters

value - true if Properties should be read only.

6.514.2.89 `virtual void activemq::commands::Message::setRecievedByDFBridge (bool recievedByDFBridge) [virtual]`

6.514.2.90 `virtual void activemq::commands::Message::setRedeliveryCounter (int redeliveryCounter) [virtual]`

6.514.2.91 `virtual void activemq::commands::Message::setReplyTo (const Pointer< ActiveMQDestination > & replyTo) [virtual]`

6.514.2.92 `virtual void activemq::commands::Message::setTargetConsumerId (const Pointer< ConsumerId > & targetConsumerId) [virtual]`

6.514.2.93 `virtual void activemq::commands::Message::setTimestamp (long long timestamp) [virtual]`

6.514.2.94 `virtual void activemq::commands::Message::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`

6.514.2.95 `virtual void activemq::commands::Message::setType (const std::string & type) [virtual]`

6.514.2.96 `virtual void activemq::commands::Message::setUserID (const std::string & userID) [virtual]`

6.514.2.97 `virtual std::string activemq::commands::Message::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 170), **activemq::commands::ActiveMQBytesMessage** (p. 205), **activemq::commands::ActiveMQMapMessage** (p. 328), **activemq::commands::ActiveMQMessage** (p. 355), **activemq::commands::ActiveMQObjectMessage**

(p. 398), `activemq::commands::ActiveMQStreamMessage` (p. 495), and `activemq::commands::ActiveMQTextMessage` (p. 609).

```
6.514.2.98 virtual Pointer<Command> activemq::commands::Message::visit  
          ( activemq::state::CommandVisitor * visitor ) throw (   
            exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1112).

6.514.3 Field Documentation

- 6.514.3.1 `long long activemq::commands::Message::arrival` [protected]
- 6.514.3.2 `long long activemq::commands::Message::brokerInTime` [protected]
- 6.514.3.3 `long long activemq::commands::Message::brokerOutTime` [protected]
- 6.514.3.4 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::Message::brokerPath` [protected]
- 6.514.3.5 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::Message::cluster` [protected]
- 6.514.3.6 `bool activemq::commands::Message::compressed` [protected]
- 6.514.3.7 `core::ActiveMQConnection* activemq::commands::Message::connection` [protected]
- 6.514.3.8 `std::vector<unsigned char> activemq::commands::Message::content` [protected]
- 6.514.3.9 `std::string activemq::commands::Message::correlationId` [protected]
- 6.514.3.10 `Pointer<DataStructure> activemq::commands::Message::dataStructure` [protected]
- 6.514.3.11 `const unsigned int activemq::commands::Message::DEFAULT_MESSAGE_SIZE = 1024` [static, protected]
- 6.514.3.12 `Pointer<ActiveMQDestination> activemq::commands::Message::destination` [protected]
- 6.514.3.13 `bool activemq::commands::Message::droppable` [protected]
- 6.514.3.14 `long long activemq::commands::Message::expiration` [protected]
- 6.514.3.15 `std::string activemq::commands::Message::groupID` [protected]
- 6.514.3.16 `int activemq::commands::Message::groupSequence` [protected]
- 6.514.3.17 `const unsigned char activemq::commands::Message::ID_MESSAGE = 0` [static]
- 6.514.3.18 `std::vector<unsigned char> activemq::commands::Message::marshalledProperties` [protected]
- 6.514.3.19 `Pointer<MessageId> activemq::commands::Message::messageId` [protected]
- 6.514.3.20 `Pointer<ActiveMQDestination> activemq::commands::Message::originalDestination` [protected]
- 6.514.3.21 `Pointer<TransactionId> activemq::commands::Message::originalTransactionId` [protected]
- 6.514.3.22 `bool activemq::commands::Message::persistent` [protected]
- 6.514.3.23 `unsigned char activemq::commands::Message::priority` [protected]

- src/main/activemq/commands/Message.h

6.515 cms::Message Class Reference

Root of all messages.

```
#include <src/main/cms/Message.h>
```

Inheritance diagram for cms::Message:

Public Member Functions

- virtual **~Message** ()
- virtual **Message * clone** () const =0
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual void **acknowledge** () const =0 throw (IllegalStateException, CMSException)
Acknowledges all consumed messages of the session of this consumed message.
- virtual void **clearBody** ()=0 throw (CMSException)
Clears out the body of the message.
- virtual void **clearProperties** ()=0 throw (CMSException)
Clears out the message body.
- virtual std::vector< std::string > **getPropertyNames** () const =0 throw (CMSException)
Retrieves the property names.
- virtual bool **propertyExists** (const std::string &name) const =0 throw (CMSException)
Indicates whether or not a given property exists.
- virtual bool **getBooleanProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSException)
Gets a boolean property.
- virtual unsigned char **getByteProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSException)
Gets a byte property.
- virtual double **getDoubleProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSException)
Gets a double property.
- virtual float **getFloatProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSException)
Gets a float property.

- virtual int **getIntProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a int property.
- virtual long long **getLongProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a long property.
- virtual short **getShortProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a short property.
- virtual std::string **getStringProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a string property.
- virtual void **setBooleanProperty** (const std::string &name, bool value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a boolean property.
- virtual void **setByteProperty** (const std::string &name, unsigned char value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a byte property.
- virtual void **setDoubleProperty** (const std::string &name, double value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a double property.
- virtual void **setFloatProperty** (const std::string &name, float value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a float property.
- virtual void **setIntProperty** (const std::string &name, int value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a int property.
- virtual void **setLongProperty** (const std::string &name, long long value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a long property.
- virtual void **setShortProperty** (const std::string &name, short value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a short property.
- virtual void **setStringProperty** (const std::string &name, const std::string &value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a string property.
- virtual std::string **getCMSCorrelationID** () const =0 throw (CMSEException)
Gets the correlation ID for the message.

- virtual void **setCMSCorrelationID** (const std::string &correlationId)=0 throw (CMSEception)
Sets the correlation ID for the message.
- virtual int **getCMSDeliveryMode** () const =0 throw (CMSEception)
*Gets the **DeliveryMode** (p. 1609) for this message.*
- virtual void **setCMSDeliveryMode** (int mode)=0 throw (CMSEception)
*Sets the **DeliveryMode** (p. 1609) for this message.*
- virtual const **Destination** * **getCMSDestination** () const =0 throw (CMSEception)
*Gets the **Destination** (p. 1610) object for this message.*
- virtual void **setCMSDestination** (const **Destination** *destination)=0 throw (CMSEception)
*Sets the **Destination** (p. 1610) object for this message.*
- virtual long long **getCMSExpiration** () const =0 throw (CMSEception)
Gets the message's expiration value.
- virtual void **setCMSExpiration** (long long expireTime)=0 throw (CMSEception)
Sets the message's expiration value.
- virtual std::string **getCMSMessageID** () const =0 throw (CMSEception)
The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.
- virtual void **setCMSMessageID** (const std::string &id)=0 throw (CMSEception)
Sets the message ID.
- virtual int **getCMSPriority** () const =0 throw (CMSEception)
Gets the message priority level.
- virtual void **setCMSPriority** (int priority)=0 throw (CMSEception)
Sets the Priority Value for this message.
- virtual bool **getCMSRedelivered** () const =0 throw (CMSEception)
Gets an indication of whether this message is being redelivered.
- virtual void **setCMSRedelivered** (bool redelivered)=0 throw (CMSEception)
Specifies whether this message is being redelivered.
- virtual const **cms::Destination** * **getCMSReplyTo** () const =0 throw (CMSEception)
*Gets the **Destination** (p. 1610) object to which a reply to this message should be sent.*
- virtual void **setCMSReplyTo** (const **cms::Destination** *destination)=0 throw (CMSEception)
*Sets the **Destination** (p. 1610) object to which a reply to this message should be sent.*

- virtual long long **getCMSTimestamp** () const =0 throw (CMSEException)
Gets the message timestamp.
- virtual void **setCMSTimestamp** (long long timeStamp)=0 throw (CMSEException)
Sets the message timestamp.
- virtual std::string **getCMSType** () const =0 throw (CMSEException)
Gets the message type identifier supplied by the client when the message was sent.
- virtual void **setCMSType** (const std::string &type)=0 throw (CMSEException)
Sets the message type.

6.515.1 Detailed Description

Root of all messages. As in JMS, a message is comprised of 3 parts: CMS-specific headers, user-defined properties, and the body.

Message (p. 2375) Bodies

The CMS API defines four types of message bodies, each type is contained within its own **Message** (p. 2375) Interface definition.

- Stream - A **StreamMessage** (p. 3415) object's message body contains a stream of primitive values in the C++ language. It is filled and read sequentially. Unlike the **BytesMessage** (p. 979) type the values written to a **StreamMessage** (p. 3415) retain information on their type and rules for type conversion are enforced when reading back the values from the **Message** (p. 2375) Body.
- Map - A **MapMessage** (p. 2318) object's message body contains a set of name-value pairs, where names are std::string objects, and values are C++ primitives. The entries can be accessed sequentially or randomly by name. The **MapMessage** (p. 2318) makes no guarantee on the order of the elements within the **Message** (p. 2375) body.
- Text - A **TextMessage** (p. 3519) object's message body contains a std::string object. This message type can be used to transport plain-text messages, and XML messages.
- Bytes - A **BytesMessage** (p. 979) object's message body contains a stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format. In many cases, it is possible to use one of the other body types, which are easier to use.

Message (p. 2375) Properties

Message (p. 2375) properties support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 1074). The String-to-primitive conversions may throw a runtime exception if the primitive's valueOf method does not accept the String as a valid representation of the primitive.

A value written as the row type can be read as the column type.

		boolean	byte	short	int	long	float	double	String
boolean	X								X
byte		X	X	X	X				X

short				X	X	X			X
int					X	X			X
long						X			X
float							X	X	X
double								X	X
String		X	X	X	X	X	X	X	X

When a **Message** (p. 2375) is delivered its properties are considered to be in a read-only mode and cannot be changed. Attempting to change the value of a delivered Message's properties will result in a **CMSEException** (p. 1074) being thrown.

See also

JMS API

Since

1.0

6.515.2 Constructor & Destructor Documentation

6.515.2.1 `virtual cms::Message::~Message () [inline, virtual]`

6.515.3 Member Function Documentation

6.515.3.1 `virtual void cms::Message::acknowledge () const throw (
 IllegalStateException, CMSEException) [pure virtual]`

Acknowledges all consumed messages of the session of this consumed message.

All consumed CMS messages support the acknowledge method for use when a client has specified that its CMS session's consumed messages are to be explicitly acknowledged. By invoking acknowledge on a consumed message, a client acknowledges all messages consumed by the session that the message was delivered to.

Calls to acknowledge are ignored for both transacted sessions and sessions specified to use implicit acknowledgment modes.

A client may individually acknowledge each message as it is consumed, or it may choose to acknowledge messages as an application-defined group (which is done by calling acknowledge on the last received message of the group, thereby acknowledging all messages consumed by the session.)

Messages that have been received but not acknowledged may be redelivered.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

IllegalStateException (p. 1868) - if this method is called on a closed session.

6.515.3.2 `virtual void cms::Message::clearBody () throw (CMSEException)
 [pure virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.3 `virtual void cms::Message::clearProperties () throw (CMSEException)`
[pure virtual]

Clears out the message body.

Clearing a message's body does not clear its header values or property entries.

If this message body was read-only, calling this method leaves the message body in the same state as an empty body in a newly created message.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.4 `virtual Message* cms::Message::clone () const` [pure virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

Implemented in `cms::BytesMessage` (p. 982).

6.515.3.5 `virtual bool cms::Message::getBooleanProperty (const std::string & name) const throw (MessageFormatException, CMSEException)` [pure virtual]

Gets a boolean property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 1074) if the property does not exist.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.515.3.6 `virtual unsigned char cms::Message::getByteProperty (const std::string & name) const throw (MessageFormatException, CMSEException)`
[pure virtual]

Gets a byte property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 1074) if the property does not exist.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.515.3.7 `virtual std::string cms::Message::getCMSCorrelationID () const throw (CMSEException) [pure virtual]`

Gets the correlation ID for the message.

This method is used to return correlation ID values that are either provider-specific message IDs or application-specific String values.

Returns

string representation of the correlation Id

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.8 `virtual int cms::Message::getCMSDeliveryMode () const throw (CMSEException) [pure virtual]`

Gets the **DeliveryMode** (p. 1609) for this message.

Returns

DeliveryMode (p. 1609) enumerated value.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.9 `virtual const Destination* cms::Message::getCMSDestination () const throw (CMSEException) [pure virtual]`

Gets the **Destination** (p. 1610) object for this message.

The CMSDestination header field contains the destination to which the message is being sent.

When a message is sent, this field is ignored. After completion of the send or publish method, the field holds the destination specified by the method.

When a message is received, its CMSDestination value must be equivalent to the value assigned when it was sent.

Returns

Destination (p. 1610) object

Exceptions

CMSException (p. 1074) - if an internal error occurs.

6.515.3.10 `virtual long long cms::Message::getCMSExpiration () const throw (CMSException) [pure virtual]`

Gets the message's expiration value.

When a message is sent, the CMSExpiration header field is left unassigned. After completion of the send or publish method, it holds the expiration time of the message. This is the sum of the time-to-live value specified by the client and the GMT at the time of the send or publish.

If the time-to-live is specified as zero, CMSExpiration is set to zero to indicate that the message does not expire.

When a message's expiration time is reached, a provider should discard it. The CMS API does not define any form of notification of message expiration.

Clients should not receive messages that have expired; however, the CMS API does not guarantee that this will not happen.

Returns

the time the message expires, which is the sum of the time-to-live value specified by the client and the GMT at the time of the send

Exceptions

CMSException (p. 1074) - if an internal error occurs.

6.515.3.11 `virtual std::string cms::Message::getCMSMessageID () const throw (CMSException) [pure virtual]`

The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.

When a message is sent, CMSMessageID can be ignored. When the send or publish method returns, it contains a provider-assigned value.

A CMSMessageID is a String value that should function as a unique key for identifying messages in a historical repository. The exact scope of uniqueness is provider-defined. It should at least cover all messages for a specific installation of a provider, where an installation is some connected set of message routers.

All CMSMessageID values must start with the prefix 'ID:'. Uniqueness of message ID values across different providers is not required.

Since message IDs take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the message ID is not used by an application. By calling the **MessageProducer.setDisableMessageID** (p. 2556) method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.

Returns

provider-assigned message id

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.12 `virtual int cms::Message::getCMSPriority () const throw (CMSEException) [pure virtual]`

Gets the message priority level.

The CMS API defines ten levels of priority value, with 0 as the lowest priority and 9 as the highest. In addition, clients should consider priorities 0-4 as gradations of normal priority and priorities 5-9 as gradations of expedited priority.

The CMS API does not require that a provider strictly implement priority ordering of messages; however, it should do its best to deliver expedited messages ahead of normal messages.

Returns

priority value

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.13 `virtual bool cms::Message::getCMSRedelivered () const throw (CMSEException) [pure virtual]`

Gets an indication of whether this message is being redelivered.

If a client receives a message with the CMSRedelivered field set, it is likely, but not guaranteed, that this message was delivered earlier but that its receipt was not acknowledged at that time.

Returns

true if this message is being redelivered

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.14 `virtual const cms::Destination* cms::Message::getCMSReplyTo () const throw (CMSEException) [pure virtual]`

Gets the **Destination** (p. 1610) object to which a reply to this message should be sent.

Returns

Destination (p. 1610) to which to send a response to this message

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.15 `virtual long long cms::Message::getCMSTimestamp () const throw (CMSEException) [pure virtual]`

Gets the message timestamp.

The CMSTimestamp header field contains the time a message was handed off to a provider to be sent. It is not the time the message was actually transmitted, because the actual send may occur later due to transactions or other client-side queuing of messages.

When a message is sent, CMSTimestamp is ignored. When the send or publish method returns, it contains a time value somewhere in the interval between the call and the return. The value is in the format of a normal millis time value in the Java programming language.

Since timestamps take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the timestamp is not used by an application. By calling the MessageProducer.setDisableMessageTimestamp method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint, the timestamp must be set to its normal value.

Returns

the message timestamp

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.16 `virtual std::string cms::Message::getCMSType () const throw (CMSEException) [pure virtual]`

Gets the message type identifier supplied by the client when the message was sent.

Returns

the message type

See also

setCMSType (p. 2391)

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.17 `virtual double cms::Message::getDoubleProperty (const std::string & name) const throw (MessageFormatException, CMSEException) [pure virtual]`

Gets a double property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 1074) if the property does not exist.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.515.3.18 `virtual float cms::Message::getFloatProperty (const std::string & name) const throw (MessageFormatException, CMSEException)`
[pure virtual]

Gets a float property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 1074) if the property does not exist.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.515.3.19 `virtual int cms::Message::getIntProperty (const std::string & name) const throw (MessageFormatException, CMSEException)`
[pure virtual]

Gets a int property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 1074) if the property does not exist.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.515.3.20 `virtual long long cms::Message::getLongProperty (const std::string & name) const throw (MessageFormatException, CMSEException)`
[pure virtual]

Gets a long property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 1074) if the property does not exist.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.515.3.21 `virtual std::vector<std::string> cms::Message::getPropertyNames ()
const throw (CMSEException) [pure virtual]`

Retrieves the property names.

Returns

The complete set of property names currently in this message.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.22 `virtual short cms::Message::getShortProperty (const std::string &
name) const throw (MessageFormatException, CMSEException)
[pure virtual]`

Gets a short property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 1074) if the property does not exist.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.515.3.23 `virtual std::string cms::Message::getStringProperty (const std::string
& name) const throw (MessageFormatException, CMSEException)
[pure virtual]`

Gets a string property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 1074) if the property does not exist.

MessageFormatException (p. 2493) - if this type conversion is invalid.

6.515.3.24 `virtual bool cms::Message::propertyExists (const std::string & name) const throw (CMSEException) [pure virtual]`

Indicates whether or not a given property exists.

Parameters

name The name of the property to look up.

Returns

True if the property exists in this message.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.25 `virtual void cms::Message::setBooleanProperty (const std::string & name, bool value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a boolean property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 1074) - if the name is an empty string.

MessageNotWriteableException (p. 2549) - if properties are read-only

6.515.3.26 `virtual void cms::Message::setByteProperty (const std::string & name, unsigned char value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a byte property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 1074) - if the name is an empty string.

MessageNotWriteableException (p. 2549) - if properties are read-only

6.515.3.27 `virtual void cms::Message::setCMSCorrelationID (const std::string & correlationId) throw (CMSEException) [pure virtual]`

Sets the correlation ID for the message.

A client can use the CMSCorrelationID header field to link one message with another. A typical use is to link a response message with its request message.

CMSCorrelationID can hold one of the following:

- A provider-specific message ID
- An application-specific String
- A provider-native byte[] value

Since each message sent by a CMS provider is assigned a message ID value, it is convenient to link messages via message ID. All message ID values must start with the 'ID:' prefix.

In some cases, an application (made up of several clients) needs to use an application-specific value for linking messages. For instance, an application may use CMSCorrelationID to hold a value referencing some external information. Application-specified values must not start with the 'ID:' prefix; this is reserved for provider-generated message ID values.

If a provider supports the native concept of correlation ID, a CMS client may need to assign specific CMSCorrelationID values to match those expected by clients that do not use the CMS API. A byte[] value is used for this purpose. CMS providers without native correlation ID values are not required to support byte[] values. The use of a byte[] value for CMSCorrelationID is non-portable.

Parameters

correlationId The message ID of a message being referred to.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.28 `virtual void cms::Message::setCMSDeliveryMode (int mode) throw (CMSEException) [pure virtual]`

Sets the **DeliveryMode** (p. 1609) for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

mode **DeliveryMode** (p. 1609) enumerated value.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.29 `virtual void cms::Message::setCMSDestination (const Destination *
destination) throw (CMSEException) [pure virtual]`

Sets the **Destination** (p. 1610) object for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

destination **Destination** (p. 1610) Object

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.30 `virtual void cms::Message::setCMSExpiration (long long expireTime
) throw (CMSEException) [pure virtual]`

Sets the message's expiration value.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

expireTime the message's expiration time

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.31 `virtual void cms::Message::setCMSMessageID (const std::string & id
) throw (CMSEException) [pure virtual]`

Sets the message ID.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

id the ID of the message

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.32 `virtual void cms::Message::setCMSPriority (int priority) throw (CMSEException) [pure virtual]`

Sets the Priority Value for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

priority priority value for this message

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.33 `virtual void cms::Message::setCMSRedelivered (bool redelivered) throw (CMSEException) [pure virtual]`

Specifies whether this message is being redelivered.

This field is set at the time the message is delivered. This method can be used to change the value for a message that has been received.

Parameters

redelivered boolean redelivered value

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.34 `virtual void cms::Message::setCMSReplyTo (const cms::Destination * destination) throw (CMSEException) [pure virtual]`

Sets the **Destination** (p. 1610) object to which a reply to this message should be sent.

The CMSReplyTo header field contains the destination where a reply to the current message should be sent. If it is null, no reply is expected. The destination may be either a **Queue** (p. 2947) object or a **Topic** (p. 3568) object.

Messages sent with a null CMSReplyTo value may be a notification of some event, or they may just be some data the sender thinks is of interest.

Messages with a CMSReplyTo value typically expect a response. A response is optional; it is up to the client to decide. These messages are called requests. A message sent in response to a request is called a reply.

In some cases a client may wish to match a request it sent earlier with a reply it has just received. The client can use the CMSCorrelationID header field for this purpose.

Parameters

destination **Destination** (p. 1610) to which to send a response to this message

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

6.515.3.35 `virtual void cms::Message::setCMSTimestamp (long long timeStamp) throw (CMSException)` [pure virtual]

Sets the message timestamp.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

timeStamp integer time stamp value

Exceptions

CMSException (p. 1074) - if an internal error occurs.

6.515.3.36 `virtual void cms::Message::setCMSType (const std::string & type) throw (CMSException)` [pure virtual]

Sets the message type.

Some CMS providers use a message repository that contains the definitions of messages sent by applications. The CMSType header field may reference a message's definition in the provider's repository.

The CMS API does not define a standard message definition repository, nor does it define a naming policy for the definitions it contains.

Some messaging systems require that a message type definition for each application message be created and that each message specify its type. In order to work with such CMS providers, CMS clients should assign a value to CMSType, whether the application makes use of it or not. This ensures that the field is properly set for those providers that require it.

To ensure portability, CMS clients should use symbolic values for CMSType that can be configured at installation time to the values defined in the current provider's message repository. If string literals are used, they may not be valid type names for some CMS providers.

Parameters

type the message type

See also

`getCMSType` (p. 2384)

Exceptions

CMSException (p. 1074) - if an internal error occurs.

6.515.3.37 `virtual void cms::Message::setDoubleProperty (const std::string & name, double value) throw (MessageNotWriteableException, CMSException)` [pure virtual]

Sets a double property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 1074) - if the name is an empty string.

MessageNotWriteableException (p. 2549) - if properties are read-only

6.515.3.38 `virtual void cms::Message::setFloatProperty (const std::string & name, float value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a float property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 1074) - if the name is an empty string.

MessageNotWriteableException (p. 2549) - if properties are read-only

6.515.3.39 `virtual void cms::Message::setIntProperty (const std::string & name, int value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a int property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 1074) - if the name is an empty string.

MessageNotWriteableException (p. 2549) - if properties are read-only

6.515.3.40 `virtual void cms::Message::setLongProperty (const std::string & name, long long value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a long property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 1074) - if the name is an empty string.

MessageNotWriteableException (p. 2549) - if properties are read-only

6.515.3.41 `virtual void cms::Message::setShortProperty (const std::string & name, short value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a short property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 1074) - if the name is an empty string.

MessageNotWriteableException (p. 2549) - if properties are read-only

6.515.3.42 `virtual void cms::Message::setStringProperty (const std::string & name, const std::string & value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a string property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 1074) - if the name is an empty string.

MessageNotWriteableException (p. 2549) - if properties are read-only

The documentation for this class was generated from the following file:

- `src/main/cms/Message.h`

6.516 activemq::commands::MessageAck Class Reference

```
#include <src/main/activemq/commands/MessageAck.h>
```

Inheritance diagram for `activemq::commands::MessageAck`:

Public Member Functions

- **MessageAck** ()
- virtual **~MessageAck** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **MessageAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual unsigned char **getAckType** () const
- virtual void **setAckType** (unsigned char ackType)
- virtual const **Pointer**< **MessageId** > & **getFirstMessageId** () const
- virtual **Pointer**< **MessageId** > & **getFirstMessageId** ()
- virtual void **setFirstMessageId** (const **Pointer**< **MessageId** > &firstMessageId)
- virtual const **Pointer**< **MessageId** > & **getLastMessageId** () const
- virtual **Pointer**< **MessageId** > & **getLastMessageId** ()
- virtual void **setLastMessageId** (const **Pointer**< **MessageId** > &lastMessageId)
- virtual int **getMessageCount** () const
- virtual void **setMessageCount** (int messageCount)
- virtual bool **isMessageAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEACK** = 22

Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **TransactionId** > **transactionId**
- **Pointer**< **ConsumerId** > **consumerId**
- unsigned char **ackType**
- **Pointer**< **MessageId** > **firstMessageId**
- **Pointer**< **MessageId** > **lastMessageId**
- int **messageCount**

6.516.1 Constructor & Destructor Documentation

6.516.1.1 **activemq::commands::MessageAck::MessageAck** ()

6.516.1.2 **virtual activemq::commands::MessageAck::~MessageAck** () [virtual]

6.516.2 Member Function Documentation

6.516.2.1 **virtual MessageAck* activemq::commands::MessageAck::cloneDataStructure** () **const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.516.2.2 **virtual void activemq::commands::MessageAck::copyDataStructure** (**const DataStructure * src**) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.516.2.3 **virtual bool activemq::commands::MessageAck::equals** (**const DataStructure * value**) **const** [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

```
6.516.2.4 virtual unsigned char activemq::commands::MessageAck::getAckType (
) const [virtual]
```

```

6.516.2.5 virtual const Pointer<ConsumerId>& ac-
activemq::commands::MessageAck::getConsumerId ( )
const [virtual]

```

```
6.516.2.6 virtual Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId ( )
[virtual]
```

```
6.516.2.7 virtual unsigned char ac-
tivismq::commands::MessageAck::getDataStructureType (
) const [virtual]
```

Get the unique identifier that this object and its own Marshaller share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.516.2.8** virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination () const [virtual]
- 6.516.2.9** virtual Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination () [virtual]
- 6.516.2.10** virtual const Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId () const [virtual]
- 6.516.2.11** virtual Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId () [virtual]
- 6.516.2.12** virtual const Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId () const [virtual]
- 6.516.2.13** virtual Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId () [virtual]
- 6.516.2.14** virtual int activemq::commands::MessageAck::getMessageCount () const [virtual]
- 6.516.2.15** virtual const Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId () const [virtual]
- 6.516.2.16** virtual Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId () [virtual]
- 6.516.2.17** virtual bool activemq::commands::MessageAck::isMessageAck () const [inline, virtual]

Returns

an answer of true to the `isMessageAck()` (p. 2397) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 698).

- 6.516.2.18 `virtual void activemq::commands::MessageAck::setAckType (unsigned char ackType) [virtual]`
- 6.516.2.19 `virtual void activemq::commands::MessageAck::setConsumerId (const Pointer< ConsumerId > & consumerId) [virtual]`
- 6.516.2.20 `virtual void activemq::commands::MessageAck::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.516.2.21 `virtual void activemq::commands::MessageAck::setFirstMessageId (const Pointer< MessageId > & firstMessageId) [virtual]`
- 6.516.2.22 `virtual void activemq::commands::MessageAck::setLastMessageId (const Pointer< MessageId > & lastMessageId) [virtual]`
- 6.516.2.23 `virtual void activemq::commands::MessageAck::setMessageCount (int messageCount) [virtual]`
- 6.516.2.24 `virtual void activemq::commands::MessageAck::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`
- 6.516.2.25 `virtual std::string activemq::commands::MessageAck::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

- 6.516.2.26 `virtual Pointer<Command> activemq::commands::MessageAck::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1112).

6.516.3 Field Documentation

- 6.516.3.1 unsigned char activemq::commands::MessageAck::ackType [protected]
- 6.516.3.2 Pointer<ConsumerId> activemq::commands::MessageAck::consumerId [protected]
- 6.516.3.3 Pointer<ActiveMQDestination> activemq::commands::MessageAck::destination [protected]
- 6.516.3.4 Pointer<MessageId> activemq::commands::MessageAck::firstMessageId [protected]
- 6.516.3.5 const unsigned char activemq::commands::MessageAck::ID_ - MESSAGEACK = 22 [static]
- 6.516.3.6 Pointer<MessageId> activemq::commands::MessageAck::lastMessageId [protected]
- 6.516.3.7 int activemq::commands::MessageAck::messageCount [protected]
- 6.516.3.8 Pointer<TransactionId> activemq::commands::MessageAck::transactionId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/MessageAck.h

6.517 activemq::wireformat::openwire::marshal::v6::MessageAckMarshal Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2399).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.517.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2399). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.517.2 Constructor & Destructor Documentation

6.517.2.1 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::MessageAckMarshaller () [inline]

6.517.2.2 virtual
activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::~~MessageAckMarshaller () [inline, virtual]

6.517.3 Member Function Documentation

6.517.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.517.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::getDataSetType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.517.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataSet * dataSet, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 729).

6.517.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataSet * dataSet, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 730).

6.517.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

6.517.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

6.517.3.7 `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h`

6.518 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2403).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller**:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.518.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2403). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.518.2 Constructor & Destructor Documentation

6.518.2.1 **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::MessageAckMarshaller** () [inline]

6.518.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::~~MessageAckMarshaller () [inline, virtual]

6.518.3 Member Function Documentation

6.518.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.518.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.518.3.3 virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 736).

6.518.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 737).

6.518.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

```
6.518.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

```
6.518.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h`

6.519 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2407).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller**:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.519.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2407). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.519.2 Constructor & Destructor Documentation

6.519.2.1 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::MessageAckMarshaller () [inline]

6.519.2.2 virtual
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::~~MessageAckMarshaller () [inline, virtual]

6.519.3 Member Function Documentation

6.519.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.519.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.519.3.3 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.519.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.519.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

```
6.519.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.519.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h`

6.520 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2411).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller**:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.520.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2411). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.520.2 Constructor & Destructor Documentation

6.520.2.1 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::MessageAckMarshaller () [inline]

6.520.2.2 virtual
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::~~MessageAckMarshaller () [inline, virtual]

6.520.3 Member Function Documentation

6.520.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.520.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.520.3.3 virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.520.3.4 virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

6.520.3.5 virtual int activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

```
6.520.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.520.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h`

6.521 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2415).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller**:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.521.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2415). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.521.2 Constructor & Destructor Documentation

6.521.2.1 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::MessageAckMarshaller () [inline]

6.521.2.2 virtual
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::~~MessageAckMarshaller () [inline, virtual]

6.521.3 Member Function Documentation

6.521.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.521.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.521.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.521.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.521.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

```
6.521.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

```
6.521.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h`

6.522 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2419).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller**:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.522.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2419). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.522.2 Constructor & Destructor Documentation

6.522.2.1 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::MessageAckMarshaller () [inline]

6.522.2.2 virtual
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::~~MessageAckMarshaller () [inline, virtual]

6.522.3 Member Function Documentation

6.522.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.522.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.522.3.3 virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 722).

6.522.3.4 virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 723).

6.522.3.5 virtual int activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```
6.522.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.522.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**MessageAckMarshaller.h**

6.523 cms::MessageConsumer Class Reference

A client uses a **MessageConsumer** (p. 2423) to received messages from a destination.

#include <src/main/cms/MessageConsumer.h>

Inheritance diagram for cms::MessageConsumer:

Public Member Functions

- virtual **~MessageConsumer** ()
- virtual **Message * receive** ()=0 throw (**CMSEException**)
*Synchronously Receive a **Message** (p. 2375).*
- virtual **Message * receive** (int millisecs)=0 throw (**CMSEException**)
*Synchronously Receive a **Message** (p. 2375), time out after defined interval.*
- virtual **Message * receiveNoWait** ()=0 throw (**CMSEException**)
*Receive a **Message** (p. 2375), does not wait if there isn't a new message to read, returns NULL if nothing read.*
- virtual void **setMessageListener** (**MessageListener** *listener)=0 throw (**CMSEException**)
*Sets the **MessageListener** (p. 2522) that this class will send notifs on.*
- virtual **MessageListener * getMessageListener** () const =0 throw (**CMSEException**)
*Gets the **MessageListener** (p. 2522) that this class will send new **Message** (p. 2375) notification events to.*
- virtual std::string **getMessageSelector** () const =0 throw (cms::CMSEException)
Gets this message consumer's message selector expression.

6.523.1 Detailed Description

A client uses a **MessageConsumer** (p. 2423) to received messages from a destination. A client may either synchronously receive a message consumer's messages or have the consumer asynchronously deliver them as they arrive.

For synchronous receipt, a client can request the next message from a message consumer using one of its **receive** methods. There are several variations of **receive** that allow a client to poll or wait for the next message.

For asynchronous delivery, a client can register a **MessageListener** (p. 2522) object with a message consumer. As messages arrive at the message consumer, it delivers them by calling the **MessageListener** (p. 2522)'s **onMessage** method.

When the **MessageConsumer**'s **close** method is called the method can block while an asynchronous message delivery is in progress or until a **receive** operation completes. A blocked consumer in a **receive** call will return a Null when the **close** method is called.

See also

MessageListener (p. 2522)

Since

1.0

6.523.2 Constructor & Destructor Documentation

6.523.2.1 **virtual cms::MessageConsumer::~MessageConsumer ()** [inline, virtual]

6.523.3 Member Function Documentation

6.523.3.1 **virtual MessageListener* cms::MessageConsumer::getMessageListener ()** **const throw (CMSEException)** [pure virtual]

Gets the **MessageListener** (p. 2522) that this class will send new **Message** (p. 2375) notification events to.

Returns

The listener of messages received by this consumer

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 994).

6.523.3.2 **virtual std::string cms::MessageConsumer::getMessageSelector ()** **const throw (cms::CMSEException)** [pure virtual]

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 995).

6.523.3.3 `virtual Message* cms::MessageConsumer::receive (int millisecs)
throw (CMSEException) [pure virtual]`

Synchronously Receive a **Message** (p. 2375), time out after defined interval.

Returns null if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 995).

6.523.3.4 `virtual Message* cms::MessageConsumer::receive () throw (CMSEException) [pure virtual]`

Synchronously Receive a **Message** (p. 2375).

Returns

new message which the caller owns and must delete.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 995).

6.523.3.5 `virtual Message* cms::MessageConsumer::receiveNoWait () throw (CMSEException) [pure virtual]`

Receive a **Message** (p. 2375), does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 996).

6.523.3.6 `virtual void cms::MessageConsumer::setMessageListener (MessageListener * listener) throw (CMSEException) [pure virtual]`

Sets the **MessageListener** (p. 2522) that this class will send notifs on.

Parameters

listener The listener of messages received by this consumer.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 996).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageConsumer.h`

6.524 activemq::cmsutil::MessageCreator Class Reference

Creates the user-defined message to be sent by the `CmsTemplate` (p. 1083).

```
#include <src/main/activemq/cmsutil/MessageCreator.h>
```

Public Member Functions

- `virtual ~MessageCreator ()`
- `virtual cms::Message * createMessage (cms::Session *session)=0 throw (cms::CMSEException)`
Creates a message from the given session.

6.524.1 Detailed Description

Creates the user-defined message to be sent by the `CmsTemplate` (p. 1083).

6.524.2 Constructor & Destructor Documentation

- 6.524.2.1** `virtual activemq::cmsutil::MessageCreator::~MessageCreator ()`
`[inline, virtual]`

6.524.3 Member Function Documentation

- 6.524.3.1** `virtual cms::Message* activemq::cmsutil::MessageCreator::createMessage (cms::Session * session) throw (cms::CMSEException) [pure virtual]`

Creates a message from the given session.

Parameters

session the CMS Session

Exceptions

cms::CMSEException (p. 1074) if thrown by CMS API methods

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/MessageCreator.h

6.525 activemq::commands::MessageDispatch Class Reference

```
#include <src/main/activemq/commands/MessageDispatch.h>
```

Inheritance diagram for activemq::commands::MessageDispatch:

Public Member Functions

- **MessageDispatch** ()
- virtual **~MessageDispatch** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **MessageDispatch * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **Message** > & **getMessage** () const
- virtual **Pointer**< **Message** > & **getMessage** ()
- virtual void **setMessage** (const **Pointer**< **Message** > &message)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int redeliveryCounter)
- virtual bool **isMessageDispatch** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGE_DISPATCH** = 21

Protected Attributes

- **Pointer< ConsumerId > consumerId**
- **Pointer< ActiveMQDestination > destination**
- **Pointer< Message > message**
- **int redeliveryCounter**

6.525.1 Constructor & Destructor Documentation

6.525.1.1 `activemq::commands::MessageDispatch::MessageDispatch ()`

6.525.1.2 `virtual activemq::commands::MessageDispatch::~~MessageDispatch ()`
[virtual]

6.525.2 Member Function Documentation

6.525.2.1 `virtual MessageDispatch* activemq::commands::MessageDispatch::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.525.2.2 `virtual void activemq::commands::MessageDispatch::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.525.2.3 `virtual bool activemq::commands::MessageDispatch::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.525.2.4 virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId () const [virtual]

6.525.2.5 virtual Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId () [virtual]

6.525.2.6 virtual unsigned char activemq::commands::MessageDispatch::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

6.525.2.7 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination () const [virtual]

6.525.2.8 virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination () [virtual]

6.525.2.9 virtual const Pointer<Message>& activemq::commands::MessageDispatch::getMessage () const [virtual]

6.525.2.10 virtual Pointer<Message>& activemq::commands::MessageDispatch::getMessage () [virtual]

6.525.2.11 virtual int activemq::commands::MessageDispatch::getRedeliveryCounter () const [virtual]

6.525.2.12 virtual bool activemq::commands::MessageDispatch::isMessageDispatch () const [inline, virtual]

Returns

an answer of true to the **isMessageDispatch()** (p. 2429) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 698).

- 6.525.2.13** `virtual void activemq::commands::MessageDispatch::setConsumerId (const Pointer< ConsumerId > & consumerId) [virtual]`
- 6.525.2.14** `virtual void activemq::commands::MessageDispatch::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.525.2.15** `virtual void activemq::commands::MessageDispatch::setMessage (const Pointer< Message > & message) [virtual]`
- 6.525.2.16** `virtual void activemq::commands::MessageDispatch::setRedeliveryCounter (int redeliveryCounter) [virtual]`
- 6.525.2.17** `virtual std::string activemq::commands::MessageDispatch::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

- 6.525.2.18** `virtual Pointer<Command> activemq::commands::MessageDispatch::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1112).

6.525.3 Field Documentation

- 6.525.3.1 `Pointer<ConsumerId> activemq::commands::MessageDispatch::consumerId`
[protected]
- 6.525.3.2 `Pointer<ActiveMQDestination> activemq::commands::MessageDispatch::destination`
[protected]
- 6.525.3.3 `const unsigned char activemq::commands::MessageDispatch::ID_ -
MESSAGEDISPATCH = 21` [static]
- 6.525.3.4 `Pointer<Message> activemq::commands::MessageDispatch::message`
[protected]
- 6.525.3.5 `int activemq::commands::MessageDispatch::redeliveryCounter`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageDispatch.h`

6.526 activemq::core::MessageDispatchChannel Class Reference

```
#include <src/main/activemq/core/MessageDispatchChannel.h>
```

Inheritance diagram for `activemq::core::MessageDispatchChannel`:

Public Member Functions

- `MessageDispatchChannel ()`
- `virtual ~MessageDispatchChannel ()`
- `void enqueue (const Pointer< MessageDispatch > &message)`
Add a Message to the Channel behind all pending message.
- `void enqueueFirst (const Pointer< MessageDispatch > &message)`
Add a message to the front of the Channel.
- `bool isEmpty () const`
- `bool isClosed () const`
- `bool isRunning () const`
- `Pointer< MessageDispatch > dequeue (long long timeout) throw (exceptions::ActiveMQException)`
Used to get an enqueued message.
- `Pointer< MessageDispatch > dequeueNoWait ()`

Used to get an enqueued message if there is one queued right now.

- **Pointer< MessageDispatch > peek () const**
Peek in the Queue and return the first message in the Channel without removing it from the channel.
- **void start ()**
Starts dispatch of messages from the Channel.
- **void stop ()**
Stops dispatch of message from the Channel.
- **void close ()**
Close this channel no messages will be dispatched after this method is called.
- **void clear ()**
Clear the Channel, all pending messages are removed.
- **int size () const**
- **std::vector< Pointer< MessageDispatch > > removeAll ()**
Remove all messages that are currently in the Channel and return them as a list of Messages.
- **virtual void lock () throw (decaf::lang::exceptions::RuntimeException)**
Locks the object.
- **virtual bool tryLock () throw (decaf::lang::exceptions::RuntimeException)**
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- **virtual void unlock () throw (decaf::lang::exceptions::RuntimeException)**
Unlocks the object.
- **virtual void wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)**
Waits on a signal from this object, which is generated by a call to Notify.
- **virtual void wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)**
Waits on a signal from this object, which is generated by a call to Notify.
- **virtual void wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)**
Waits on a signal from this object, which is generated by a call to Notify.
- **virtual void notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)**
Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.526.1 Constructor & Destructor Documentation

6.526.1.1 `activemq::core::MessageDispatchChannel::MessageDispatchChannel ()`

6.526.1.2 `virtual
activemq::core::MessageDispatchChannel::~~MessageDispatchChannel () [virtual]`

6.526.2 Member Function Documentation

6.526.2.1 `void activemq::core::MessageDispatchChannel::clear ()`

Clear the Channel, all pending messages are removed.

6.526.2.2 `void activemq::core::MessageDispatchChannel::close ()`

Close this channel no messages will be dispatched after this method is called.

6.526.2.3 `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeue (long long timeout) throw (exceptions::ActiveMQException)`

Used to get an enqueued message.

The amount of time this method blocks is based on the timeout value. - if timeout==-1 then it blocks until a message is received. - if timeout==0 then it tries to not block at all, it returns a message if it is available - if timeout>0 then it blocks up to timeout amount of time. Expired messages will be consumed by this method.

Returns

null if we timeout or if the consumer is closed.

Exceptions

ActiveMQException

6.526.2.4 `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeueNoWait ()`

Used to get an enqueued message if there is one queued right now.

If there is no waiting message then this method returns Null.

Returns

a message if there is one in the queue.

6.526.2.5 `void activemq::core::MessageDispatchChannel::enqueue (const Pointer< MessageDispatch > & message)`

Add a Message to the Channel behind all pending message.

Parameters

message - The message to add to the Channel.

6.526.2.6 `void activemq::core::MessageDispatchChannel::enqueueFirst (const Pointer< MessageDispatch > & message)`

Add a message to the front of the Channel.

Parameters

message - The Message to add to the front of the Channel.

6.526.2.7 `bool activemq::core::MessageDispatchChannel::isClosed () const [inline]`

Returns

has the Queue been closed.

6.526.2.8 `bool activemq::core::MessageDispatchChannel::isEmpty () const`

Returns

true if there are no messages in the Channel.

6.526.2.9 `bool activemq::core::MessageDispatchChannel::isRunning () const [inline]`

Returns

true if the Channel currently running and will dequeue message.

6.526.2.10 `virtual void activemq::core::MessageDispatchChannel::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3463).

6.526.2.11 `virtual void activemq::core::MessageDispatchChannel::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3464).

6.526.2.12 `virtual void activemq::core::MessageDispatchChannel::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3465).

6.526.2.13 `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::peek () const`

Peek in the Queue and return the first message in the Channel without removing it from the channel.

Returns

a message if there is one in the queue.

6.526.2.14 `std::vector< Pointer<MessageDispatch> > activemq::core::MessageDispatchChannel::removeAll ()`

Remove all messages that are currently in the Channel and return them as a list of Messages.

Returns

a list of Messages that was previously in the Channel.

6.526.2.15 `int activemq::core::MessageDispatchChannel::size () const`

Returns

the number of Messages currently in the Channel.

6.526.2.16 `void activemq::core::MessageDispatchChannel::start ()`

Starts dispatch of messages from the Channel.

6.526.2.17 `void activemq::core::MessageDispatchChannel::stop ()`

Stops dispatch of message from the Channel.

6.526.2.18 `virtual bool activemq::core::MessageDispatchChannel::tryLock ()
throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3466).

6.526.2.19 `virtual void activemq::core::MessageDispatchChannel::unlock ()
throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3467).

6.526.2.20 `virtual void activemq::core::MessageDispatchChannel::wait (long
long milliseconds) throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3470).

```
6.526.2.21  virtual void activemq::core::MessageDispatchChannel::wait
            ( long long millisecs, int nanos ) throw
            ( decaf::lang::exceptions::RuntimeException,
              decaf::lang::exceptions::IllegalArgumentException,
              decaf::lang::exceptions::IllegalMonitorStateException,
              decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3471).

```
6.526.2.22  virtual void activemq::core::MessageDispatchChannel::wait
            ( ) throw ( decaf::lang::exceptions::RuntimeException,
              decaf::lang::exceptions::IllegalMonitorStateException,
              decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3468).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/MessageDispatchChannel.h`

6.527 activemq::wireformat::openwire::marshal::v2::MessageDispatchMa Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2438).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.527.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2438). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.527.2 Constructor & Destructor Documentation

6.527.2.1 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::MessageDispatchMarshaller () [inline]

6.527.2.2 virtual
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::~~MessageDispatchMarshaller () [inline, virtual]

6.527.3 Member Function Documentation

6.527.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.527.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.527.3.3 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.527.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

6.527.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 738).

```
6.527.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 739).

```
6.527.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h`

6.528 activemq::wireformat::openwire::marshal::v3::MessageDispatchMa Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2442).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller**:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.528.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p.2442). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.528.2 Constructor & Destructor Documentation

6.528.2.1 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::MessageDispatchMarshaller () [inline]

6.528.2.2 virtual
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::~MessageDispatchMarshaller () [inline, virtual]

6.528.3 Member Function Documentation

6.528.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.528.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.528.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.528.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.528.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

```
6.528.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.528.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h`

6.529 activemq::wireformat::openwire::marshal::v5::MessageDispatchMa Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2446).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller**:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.529.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p.2446). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.529.2 Constructor & Destructor Documentation

6.529.2.1 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::MessageDispatchMarshaller () [inline]

6.529.2.2 virtual
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::~MessageDispatchMarshaller () [inline, virtual]

6.529.3 Member Function Documentation

6.529.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.529.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.529.3.3 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.529.3.4 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

6.529.3.5 virtual int activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```
6.529.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.529.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h`

6.530 activemq::wireformat::openwire::marshal::v4::MessageDispatchMa Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2450).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller**:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.530.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p.2450). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.530.2 Constructor & Destructor Documentation

6.530.2.1 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::MessageDispatchMarshaller () [inline]

6.530.2.2 virtual
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::~~MessageDispatchMarshaller () [inline, virtual]

6.530.3 Member Function Documentation

6.530.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.530.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.530.3.3 virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.530.3.4 virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

6.530.3.5 virtual int activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 711).

```
6.530.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 712).

```
6.530.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h`

6.531 activemq::wireformat::openwire::marshal::v1::MessageDispatchMa Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2454).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller**:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.531.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p.2454). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.531.2 Constructor & Destructor Documentation

6.531.2.1 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::MessageDispatchMarshaller () [inline]

6.531.2.2 virtual
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::~MessageDispatchMarshaller () [inline, virtual]

6.531.3 Member Function Documentation

6.531.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.531.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.531.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.531.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.531.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 718).

```
6.531.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 719).

```
6.531.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h`

6.532 **activemq::wireformat::openwire::marshal::v6::MessageDispatchMa** Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2458).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller**:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.532.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p.2458). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.532.2 Constructor & Destructor Documentation

6.532.2.1 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::MessageDispatchMarshaller () [inline]

6.532.2.2 virtual activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::~MessageDispatchMarshaller () [inline, virtual]

6.532.3 Member Function Documentation

6.532.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.532.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.532.3.3 virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.532.3.4 virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

6.532.3.5 virtual int activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 731).

```
6.532.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 732).

```
6.532.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h`

6.533 activemq::commands::MessageDispatchNotification Class Reference

```
#include <src/main/activemq/commands/MessageDispatchNotification.h>
```

Inheritance diagram for `activemq::commands::MessageDispatchNotification`:

Public Member Functions

- **MessageDispatchNotification** ()
- virtual **~MessageDispatchNotification** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **MessageDispatchNotification * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual long long **getDeliverySequenceId** () const
- virtual void **setDeliverySequenceId** (long long deliverySequenceId)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const

- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)
- virtual bool **isMessageDispatchNotification** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** ***visitor**)
throw (**exceptions::ActiveMQException**)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _MESSAGEDISPATCHNOTIFICATION** = 90

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **deliverySequenceId**
- **Pointer**< **MessageId** > **messageId**

6.533.1 Constructor & Destructor Documentation

6.533.1.1 **activemq::commands::MessageDispatchNotification::MessageDispatchNotification**
()

6.533.1.2 **virtual**
activemq::commands::MessageDispatchNotification::~~MessageDispatchNotification
() [virtual]

6.533.2 Member Function Documentation

6.533.2.1 **virtual MessageDispatchNotification* ac-**
tivemq::commands::MessageDispatchNotification::cloneDataStructure (
) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p.1554).

6.533.2.2 **virtual void ac-**
tivemq::commands::MessageDispatchNotification::copyDataStructure (
const **DataStructure** * **src**) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.533.2.3 `virtual bool activemq::commands::MessageDispatchNotification::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.533.2.4 `virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId () const [virtual]`

6.533.2.5 `virtual Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId () [virtual]`

6.533.2.6 `virtual unsigned char activemq::commands::MessageDispatchNotification::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.533.2.7 virtual long long activemq::commands::MessageDispatchNotification::getDeliverySequenceId () const [virtual]
- 6.533.2.8 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination () const [virtual]
- 6.533.2.9 virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination () [virtual]
- 6.533.2.10 virtual Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId () [virtual]
- 6.533.2.11 virtual const Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId () const [virtual]
- 6.533.2.12 virtual bool activemq::commands::MessageDispatchNotification::isMessageDispatchNotification () const [inline, virtual]

Returns

an answer of true to the `isMessageDispatchNotification()` (p. 2465) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 699).

- 6.533.2.13 virtual void activemq::commands::MessageDispatchNotification::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]
- 6.533.2.14 virtual void activemq::commands::MessageDispatchNotification::setDeliverySequenceId (long long *deliverySequenceId*) [virtual]
- 6.533.2.15 virtual void activemq::commands::MessageDispatchNotification::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.533.2.16 virtual void activemq::commands::MessageDispatchNotification::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]
- 6.533.2.17 virtual std::string activemq::commands::MessageDispatchNotification::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 700).

6.533.2.18 `virtual Pointer<Command> activemq::commands::MessageDispatchNotification::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1112).

6.533.3 Field Documentation

6.533.3.1 `Pointer<ConsumerId> activemq::commands::MessageDispatchNotification::consumerId [protected]`

6.533.3.2 `long long activemq::commands::MessageDispatchNotification::deliverySequenceId [protected]`

6.533.3.3 `Pointer<ActiveMQDestination> activemq::commands::MessageDispatchNotification::destination [protected]`

6.533.3.4 `const unsigned char activemq::commands::MessageDispatchNotification::ID_ - MESSAGEDISPATCHNOTIFICATION = 90 [static]`

6.533.3.5 `Pointer<MessageId> activemq::commands::MessageDispatchNotification::messageId [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageDispatchNotification.h`

6.534 `activemq::wireformat::openwire::marshal::v6::MessageDispatchNo` Class Reference

Marshaling code for Open Wire Format for `MessageDispatchNotificationMarshaller` (p. 2466).

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.534.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2466). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.534.2 Constructor & Destructor Documentation

6.534.2.1 `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

6.534.2.2 `virtual activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

6.534.3 Member Function Documentation

6.534.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.534.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.534.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::marshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.534.3.4 `virtual void activismq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::looseM
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn) throw (
decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller`
(p. 730).

6.534.3.5 `virtual int activismq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightM
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs) throw (
decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller`
(p. 731).

6.534.3.6 `virtual void activismq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightM
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

```
6.534.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightU
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchNotificationMarshaller.h`

6.535 `activemq::wireformat::openwire::marshal::v2::MessageDispatchNo` Class Reference

Marshaling code for Open Wire Format for `MessageDispatchNotificationMarshaller` (p. 2470).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`:

 Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.535.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2470). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.535.2 Constructor & Destructor Documentation

6.535.2.1 `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller () [inline]`

6.535.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller () [inline, virtual]`

6.535.3 Member Function Documentation

6.535.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::create () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.535.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::getDataType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.535.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.535 ac-
 tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 Class Reference 2477

6.535.3.4 **virtual void ac-**
 tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::looseM
 (OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * *dataIn*) throw (
 decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**
 (p. 737).

6.535.3.5 **virtual int ac-**
 tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightM
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, utils::BooleanStream * *bs*) throw (
 decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**
 (p. 738).

6.535.3.6 **virtual void ac-**
 tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightM
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 739).

```
6.535.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightU
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h

6.536 activemq::wireformat::openwire::marshal::v3::MessageDispatchNo

Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2474).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**:

 Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.536.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2474). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.536.2 Constructor & Destructor Documentation

6.536.2.1 `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller () [inline]`

6.536.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller () [inline, virtual]`

6.536.3 Member Function Documentation

6.536.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::create () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.536.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.536.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.536 ac-
tivismq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
Class Reference 2481

6.536.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::looseM
(OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
(p. 703).

6.536.3.5 virtual int ac-
tivismq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightM
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, utils::BooleanStream * *bs*) throw (
decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
(p. 704).

6.536.3.6 virtual void ac-
tivismq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightM
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 706).

```
6.536.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightU
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h

6.537 activemq::wireformat::openwire::marshal::v4::MessageDispatchNo Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2478).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**:

 Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.537.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2478). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.537.2 Constructor & Destructor Documentation

6.537.2.1 `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller () [inline]`

6.537.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller () [inline, virtual]`

6.537.3 Member Function Documentation

6.537.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::create () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.537.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.537.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.537 ac-
 tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller
 Class Reference 2485

6.537.3.4 virtual void ac-
 tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::looseM
 (OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * *dataIn*) throw (
 decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**
 (p. 710).

6.537.3.5 virtual int ac-
 tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightM
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, utils::BooleanStream * *bs*) throw (
 decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**
 (p. 711).

6.537.3.6 virtual void ac-
 tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightM
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 712).

```
6.537.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightU
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h

6.538 activemq::wireformat::openwire::marshal::v1::MessageDispatchNo Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2482).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**:

 Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.538.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2482). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.538.2 Constructor & Destructor Documentation

6.538.2.1 `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller () [inline]`

6.538.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller () [inline, virtual]`

6.538.3 Member Function Documentation

6.538.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::createDataStructure () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.538.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.538.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.538 ac-
 tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 Class Reference 2489

6.538.3.4 **virtual void ac-**
 tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::looseM
 (OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * *dataIn*) throw (
 decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**
 (p. 717).

6.538.3.5 **virtual int ac-**
 tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightM
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, utils::BooleanStream * *bs*) throw (
 decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**
 (p. 718).

6.538.3.6 **virtual void ac-**
 tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightM
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 719).

```
6.538.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightU
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h

6.539 activemq::wireformat::openwire::marshal::v5::MessageDispatchNo

Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2486).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**:

 Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.539.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2486). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.539.2 Constructor & Destructor Documentation

6.539.2.1 `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller () [inline]`

6.539.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller () [inline, virtual]`

6.539.3 Member Function Documentation

6.539.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::create () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.539.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::getDataType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.539.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.539 ac-
tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
Class Reference **2493**

6.539.3.4 `virtual void ac-`
tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::looseM
(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**
 (p. 723).

6.539.3.5 `virtual int ac-`
tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightM
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, utils::BooleanStream * *bs*) throw (**
decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**
 (p. 724).

6.539.3.6 `virtual void ac-`
tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightM
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataOutputStream * *dataOut*,**
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 726).

```
6.539.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightU
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h

6.540 cms::MessageEnumeration Class Reference

Defines an object that enumerates a collection of Messages.

```
#include <src/main/cms/MessageEnumeration.h>
```

Inheritance diagram for cms::MessageEnumeration:

Public Member Functions

- virtual `~MessageEnumeration ()`
- virtual bool `hasMoreMessages ()=0`
*Returns true if there are more **Message** (p. 2375) in the Browser that can be retrieved via the **nextMessage** method.*
- virtual `cms::Message * nextMessage ()=0 throw (cms::CMSEException)`
*Returns the Next **Message** (p. 2375) in the **Queue** (p. 2947) if one is present, if no more Message's are available then an Exception is thrown.*

6.540.1 Detailed Description

Defines an object that enumerates a collection of Messages. The client calls the `hasMoreMessages` method to determine if a **Message** (p. 2375) is available. If a **Message** (p. 2375) is available the client calls the `nextMessage` method to retrieve that **Message** (p. 2375), calling `nextMessage` when a **Message** (p. 2375) is not available results in an exception.

Since

2.1

6.540.2 Constructor & Destructor Documentation

- 6.540.2.1** virtual `cms::MessageEnumeration::~~MessageEnumeration ()` [inline, virtual]

6.540.3 Member Function Documentation

- 6.540.3.1** virtual bool `cms::MessageEnumeration::hasMoreMessages ()` [pure virtual]

Returns true if there are more **Message** (p. 2375) in the Browser that can be retrieved via the `nextMessage` method.

If this method returns false and the `nextMessage` method is called then an Exception will be thrown.

Returns

true if more Message's are available in the Browser.

Implemented in `activemq::core::ActiveMQQueueBrowser` (p. 440).

- 6.540.3.2** virtual `cms::Message* cms::MessageEnumeration::nextMessage ()`
`throw (cms::CMSEException)` [pure virtual]

Returns the Next **Message** (p. 2375) in the **Queue** (p. 2947) if one is present, if no more Message's are available then an Exception is thrown.

If a **Message** (p. 2375) object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

Returns

The next **Message** (p. 2375) in the **Queue** (p. 2947).

Exceptions

CMSEException (p. 1074) if no more Message's currently in the **Queue** (p. 2947).

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 440).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageEnumeration.h`

6.541 cms::MessageEOFException Class Reference

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3415) or **BytesMessage** (p. 979) is being read.

```
#include <src/main/cms/MessageEOFException.h>
```

Inheritance diagram for cms::MessageEOFException:

Public Member Functions

- **MessageEOFException** () throw ()
- **MessageEOFException** (const **MessageEOFException** &ex) throw ()
- **MessageEOFException** (const std::string &message, const std::exception *cause) throw ()
- **MessageEOFException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageEOFException** () throw ()

6.541.1 Detailed Description

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3415) or **BytesMessage** (p. 979) is being read.

Since

1.3

6.541.2 Constructor & Destructor Documentation

- 6.541.2.1 `cms::MessageEOFException::MessageEOFException () throw ()`
- 6.541.2.2 `cms::MessageEOFException::MessageEOFException (const MessageEOFException & ex) throw ()`
- 6.541.2.3 `cms::MessageEOFException::MessageEOFException (const std::string & message, const std::exception * cause) throw ()`
- 6.541.2.4 `cms::MessageEOFException::MessageEOFException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.541.2.5 `virtual cms::MessageEOFException::~MessageEOFException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageEOFException.h`

6.542 cms::MessageFormatException Class Reference

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

```
#include <src/main/cms/MessageFormatException.h>
```

Inheritance diagram for cms::MessageFormatException:

Public Member Functions

- `MessageFormatException () throw ()`
- `MessageFormatException (const MessageFormatException &ex) throw ()`
- `MessageFormatException (const std::string &message, const std::exception *cause) throw ()`
- `MessageFormatException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~MessageFormatException () throw ()`

6.542.1 Detailed Description

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type. It must also be thrown when equivalent type errors are made with message property values. For example, this exception must be thrown if `StreamMessage.readShort` (p. 3422) is used to read a boolean value.

Since

1.3

6.542.2 Constructor & Destructor Documentation

- 6.542.2.1 `cms::MessageFormatException::MessageFormatException () throw ()`
- 6.542.2.2 `cms::MessageFormatException::MessageFormatException (const MessageFormatException & ex) throw ()`
- 6.542.2.3 `cms::MessageFormatException::MessageFormatException (const std::string & message, const std::exception * cause) throw ()`
- 6.542.2.4 `cms::MessageFormatException::MessageFormatException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.542.2.5 `virtual cms::MessageFormatException::~MessageFormatException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageFormatException.h`

6.543 activemq::commands::MessageId Class Reference

```
#include <src/main/activemq/commands/MessageId.h>
```

Inheritance diagram for `activemq::commands::MessageId`:

Public Types

- `typedef decaf::lang::PointerComparator< MessageId > COMPARATOR`

Public Member Functions

- `MessageId ()`
- `MessageId (const MessageId &other)`
- `MessageId (const std::string &messageKey)`
- `MessageId (const Pointer< ProducerInfo > &producerInfo, long long producerSequenceId)`
- `MessageId (const Pointer< ProducerId > &producerId, long long producerSequenceId)`
- `MessageId (const std::string &producerId, long long producerSequenceId)`
- `virtual ~MessageId ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual MessageId * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

- void **setValue** (const std::string &key)
- void **setTextView** (const std::string &key)
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual long long **getProducerSequenceId** () const
- virtual void **setProducerSequenceId** (long long producerSequenceId)
- virtual long long **getBrokerSequenceId** () const
- virtual void **setBrokerSequenceId** (long long brokerSequenceId)
- virtual int **compareTo** (const **MessageId** &value) const
- virtual bool **equals** (const **MessageId** &value) const
- virtual bool **operator==** (const **MessageId** &value) const
- virtual bool **operator<** (const **MessageId** &value) const
- **MessageId** & **operator=** (const **MessageId** &other)

Static Public Attributes

- static const unsigned char **ID_MESSAGEID** = 110

Protected Attributes

- **Pointer**< **ProducerId** > **producerId**
- long long **producerSequenceId**
- long long **brokerSequenceId**

6.543.1 Member Typedef Documentation

6.543.1.1 `typedef decaf::lang::PointerComparator<MessageId>
activemq::commands::MessageId::COMPARATOR`

6.543.2 Constructor & Destructor Documentation

6.543.2.1 `activemq::commands::MessageId::MessageId ()`

6.543.2.2 `activemq::commands::MessageId::MessageId (const MessageId & other
)`

6.543.2.3 `activemq::commands::MessageId::MessageId (const std::string &
messageKey)`

6.543.2.4 `activemq::commands::MessageId::MessageId (const Pointer<
ProducerInfo > & producerInfo, long long producerSequenceId)`

6.543.2.5 `activemq::commands::MessageId::MessageId (const Pointer<
ProducerId > & producerId, long long producerSequenceId)`

6.543.2.6 `activemq::commands::MessageId::MessageId (const std::string &
producerId, long long producerSequenceId)`

6.543.2.7 `virtual activemq::commands::MessageId::~~MessageId () [virtual]`

6.543.3 Member Function Documentation

6.543.3.1 `virtual MessageId* activemq::commands::MessageId::cloneDataStructure
() const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.543.3.2 `virtual int activemq::commands::MessageId::compareTo (const
MessageId & value) const [virtual]`

6.543.3.3 `virtual void activemq::commands::MessageId::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.543.3.4 `virtual bool activemq::commands::MessageId::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.543.3.5 `virtual bool activemq::commands::MessageId::equals (const MessageId & value) const [virtual]`

6.543.3.6 `virtual long long activemq::commands::MessageId::getBrokerSequenceId () const [virtual]`

6.543.3.7 `virtual unsigned char activemq::commands::MessageId::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

- 6.543.3.8 `virtual const Pointer<ProducerId>& activemq::commands::MessageId::getProducerId () const`
[virtual]
- 6.543.3.9 `virtual Pointer<ProducerId>& activemq::commands::MessageId::getProducerId ()`
[virtual]
- 6.543.3.10 `virtual long long activemq::commands::MessageId::getProducerSequenceId () const` [virtual]
- 6.543.3.11 `virtual bool activemq::commands::MessageId::operator< (const MessageId & value) const` [virtual]
- 6.543.3.12 `MessageId& activemq::commands::MessageId::operator= (const MessageId & other)`
- 6.543.3.13 `virtual bool activemq::commands::MessageId::operator== (const MessageId & value) const` [virtual]
- 6.543.3.14 `virtual void activemq::commands::MessageId::setBrokerSequenceId (long long brokerSequenceId)` [virtual]
- 6.543.3.15 `virtual void activemq::commands::MessageId::setProducerId (const Pointer< ProducerId > & producerId)` [virtual]
- 6.543.3.16 `virtual void activemq::commands::MessageId::setProducerSequenceId (long long producerSequenceId)` [virtual]
- 6.543.3.17 `void activemq::commands::MessageId::setTextView (const std::string & key)`
- 6.543.3.18 `void activemq::commands::MessageId::setValue (const std::string & key)`
- 6.543.3.19 `virtual std::string activemq::commands::MessageId::toString () const`
[virtual]

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.543.4 Field Documentation

- 6.543.4.1 `long long activemq::commands::MessageId::brokerSequenceId`
[protected]
- 6.543.4.2 `const unsigned char activemq::commands::MessageId::ID_MESSAGEID = 110` [static]
- 6.543.4.3 `Pointer<ProducerId> activemq::commands::MessageId::producerId`
[protected]
- 6.543.4.4 `long long activemq::commands::MessageId::producerSequenceId`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageId.h`

6.544 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for `MessageIdMarshaller` (p. 2499).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller`:

Public Member Functions

- `MessageIdMarshaller ()`
- `virtual ~MessageIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.544.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2499). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.544.2 Constructor & Destructor Documentation

6.544.2.1 **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::MessageIdMarshaller** () [inline]

6.544.2.2 **virtual** **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::~~MessageIdMarshaller** () [inline, virtual]

6.544.3 Member Function Documentation

6.544.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.544.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.544.3.3 virtual void `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseMarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.544.3.4 virtual void `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.544.3.5 virtual int `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal1` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

```
6.544.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

```
6.544.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException*** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessageIdMarshaller.h**

6.545 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2503).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller**:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.545.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p.2503). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.545.2 Constructor & Destructor Documentation

6.545.2.1 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::MessageIdMarshaller () [inline]

6.545.2.2 virtual activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::~~MessageIdMarshaller () [inline, virtual]

6.545.3 Member Function Documentation

6.545.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.545.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.545.3.3 virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1518).

6.545.3.4 virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.545.3.5 virtual int activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

```
6.545.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

```
6.545.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h`

6.546 activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2507).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller**:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.546.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p.2507). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.546.2 Constructor & Destructor Documentation

6.546.2.1 `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::MessageIdMarshaller () [inline]`

6.546.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::~~MessageIdMarshaller () [inline, virtual]`

6.546.3 Member Function Documentation

6.546.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.546.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.546.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.546.3.4 virtual void `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.546.3.5 virtual int `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal1` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

```

6.546.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

```

6.546.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h`

6.547 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for `MessageIdMarshaller` (p. 2510).


```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.547.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p.2510). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.547.2 Constructor & Destructor Documentation

6.547.2.1 `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::MessageIdMarshaller () [inline]`

6.547.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::~~MessageIdMarshaller () [inline, virtual]`

6.547.3 Member Function Documentation

6.547.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.547.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.547.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.547.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.547.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.547.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.547.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h`

6.548 **activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2514).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller**:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.548.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p.2514). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.548.2 Constructor & Destructor Documentation

6.548.2.1 `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::MessageIdMarshaller () [inline]`

6.548.2.2 `virtual activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::~~MessageIdMarshaller () [inline, virtual]`

6.548.3 Member Function Documentation

6.548.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.548.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.548.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.548.3.4 virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.548.3.5 virtual int activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.548.3.6 virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.548.3.7 virtual void **activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h`

6.549 **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2518).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.549.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p.2518). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.549.2 Constructor & Destructor Documentation

6.549.2.1 `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::MessageIdMarshaller () [inline]`

6.549.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::~~MessageIdMarshaller () [inline, virtual]`

6.549.3 Member Function Documentation

6.549.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.549.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.549.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.549.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.549.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.549.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.549.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h`

6.550 cms::MessageListener Class Reference

A `MessageListener` (p. 2522) object is used to receive asynchronously delivered messages.

```
#include <src/main/cms/MessageListener.h>
```

Public Member Functions

- `virtual ~MessageListener ()`
- `virtual void onMessage (const Message *message)=0`

*Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 2375) types.*

6.550.1 Detailed Description

A `MessageListener` (p. 2522) object is used to receive asynchronously delivered messages.

Since

1.0

6.550.2 Constructor & Destructor Documentation

6.550.2.1 `virtual cms::MessageListener::~~MessageListener () [inline, virtual]`

6.550.3 Member Function Documentation

6.550.3.1 `virtual void cms::MessageListener::onMessage (const Message * message) [pure virtual]`

Called asynchronously when a new message is received, the message reference can be to any of the `Message` (p. 2375) types.

a dynamic cast is used to find out what type of message this is. The lifetime of this object is only guaranteed to be for life of the `onMessage` function after this call-back returns the message may no longer exists. Users should copy the data or clone the message if they wish to retain information that was contained in this `Message` (p. 2375).

It is considered a programming error for this method to throw an exception.

Parameters

message `Message` (p. 2375) object {const} pointer recipient does not own.

The documentation for this class was generated from the following file:

- `src/main/cms/MessageListener.h`

6.551 activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2523).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::MessageMarshaller`:

Public Member Functions

- `MessageMarshaller ()`
- `virtual ~MessageMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.551.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2523). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.551.2 Constructor & Destructor Documentation

6.551.2.1 activemq::wireformat::openwire::marshal::v5::MessageMarshaller::MessageMarshaller () [inline]

6.551.2.2 virtual activemq::wireformat::openwire::marshal::v5::MessageMarshaller::~~MessageMarshaller () [inline, virtual]

6.551.3 Member Function Documentation

6.551.3.1 virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 223), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 342), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 369), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 413), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 513), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 623).

6.551.3.2 virtual void `activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 189), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 224), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 343), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 369), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 413), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 514), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 623).

```

6.551.3.3 virtual int ac-
tivismq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 189), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 224), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 343), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 369), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 413), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 514), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 623).

```

6.551.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 189), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 224), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 343), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 370), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 414), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 514), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 624).

```
6.551.3.5  virtual void ac-
            tivemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataInputStream * dataIn,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 727).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 190), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 225), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 344), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 370), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 414), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 515), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 624).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h`

6.552 activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2527).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::MessageMarshaller`:

Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.552.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2527). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.552.2 Constructor & Destructor Documentation

6.552.2.1 `activemq::wireformat::openwire::marshal::v3::MessageMarshaller::MessageMarshaller ()` [inline]

6.552.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageMarshaller::~~MessageMarshaller ()` [inline, virtual]

6.552.3 Member Function Documentation

6.552.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 172), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 211), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 330), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 501), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 611).

6.552.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 173), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 212), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 502), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 611).

```
6.552.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 173), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 212), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 502), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 612).

```

6.552.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 174), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 213), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 358), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 402), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 502), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 612).

```

6.552.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 707).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 174), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 213), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 332), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 358), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 402), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 503), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 612).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h`

6.553 `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2532).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageMarshaller`:

Public Member Functions

- `MessageMarshaller ()`
- `virtual ~MessageMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.553.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2532). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.553.2 Constructor & Destructor Documentation

6.553.2.1 `activemq::wireformat::openwire::marshal::v2::MessageMarshaller::MessageMarshaller ()` [inline]

6.553.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessageMarshaller::~~MessageMarshaller ()` [inline, virtual]

6.553.3 Member Function Documentation

6.553.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 346), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 373), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 417), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 517), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 631).

6.553.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 185), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 232), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 347), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 373), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 417), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 518), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 631).

```
6.553.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 185), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 232), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 347), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 373), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 417), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 518), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 631).


```

6.553.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 185), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 232), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 347), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 374), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 418), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 518), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 632).

```

6.553.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 740).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 186), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 233), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 348), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 374), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 418), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 519), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 632).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h`

6.554 `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2536).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::MessageMarshaller`:

Public Member Functions

- `MessageMarshaller ()`
- `virtual ~MessageMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.554.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2536). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.554.2 Constructor & Destructor Documentation

6.554.2.1 `activemq::wireformat::openwire::marshal::v4::MessageMarshaller::MessageMarshaller ()` [inline]

6.554.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageMarshaller::~~MessageMarshaller ()` [inline, virtual]

6.554.3 Member Function Documentation

6.554.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 180), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 219), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 338), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 365), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 409), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 509), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 615).

6.554.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 181), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 220), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 339), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 365), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 409), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 510), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 615).

```
6.554.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 181), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 220), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 339), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 365), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 409), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 510), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 615).

```

6.554.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 181), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 220), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 339), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 366), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 410), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 510), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 616).

```

6.554.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 713).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 182), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 221), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 340), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 366), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 410), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 511), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 616).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h`

6.555 `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2540).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::MessageMarshaller`:

Public Member Functions

- `MessageMarshaller ()`
- `virtual ~MessageMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.555.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2540). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.555.2 Constructor & Destructor Documentation

6.555.2.1 `activemq::wireformat::openwire::marshal::v1::MessageMarshaller::MessageMarshaller ()` [inline]

6.555.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v1::MessageMarshaller::~~MessageMarshaller ()` [inline, virtual]

6.555.3 Member Function Documentation

6.555.3.1 `virtual void` `activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 176), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 215), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 334), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 361), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 405), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 505), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 619).

6.555.3.2 `virtual void` `activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn)` throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 177), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 216), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 335), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 361), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 405), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 506), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 619).

```
6.555.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 177), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 216), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 335), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 361), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 405), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 506), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 619).

6.555.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 719).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 177), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 216), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 335), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 362), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 406), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 506), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 620).

6.555.3.5 virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 178), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 217), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 336), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 362), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 406), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 507), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 620).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h`

6.556 `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2544).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::MessageMarshaller`:

Public Member Functions

- `MessageMarshaller ()`
- `virtual ~MessageMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.556.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2544). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.556.2 Constructor & Destructor Documentation

6.556.2.1 `activemq::wireformat::openwire::marshal::v6::MessageMarshaller::MessageMarshaller ()` [inline]

6.556.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v6::MessageMarshaller::~~MessageMarshaller ()` [inline, virtual]

6.556.3 Member Function Documentation

6.556.3.1 `virtual void` `activemq::wireformat::openwire::marshal::v6::MessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 227), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 350), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 421), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 521), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 627).

6.556.3.2 `virtual void` `activemq::wireformat::openwire::marshal::v6::MessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn)` throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 193), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 228), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 351), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 421), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 522), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 627).

```
6.556.3.3  virtual int ac-
            tivemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure
              * dataStructure, utils::BooleanStream * bs ) throw (
              decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 193), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 228), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 351), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 421), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 522), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 627).

```

6.556.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 732).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 193), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 228), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 351), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 378), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 422), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 522), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 628).

```

6.556.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 194), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 229), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 352), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 378), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 422), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 523), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 628).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h`

6.557 cms::MessageNotReadableException Class Reference

This exception must be thrown when a CMS client attempts to read a write-only message.

```
#include <src/main/cms/MessageNotReadableException.h>
```

Inheritance diagram for `cms::MessageNotReadableException`:

Public Member Functions

- `MessageNotReadableException () throw ()`
- `MessageNotReadableException (const MessageNotReadableException &ex) throw ()`
- `MessageNotReadableException (const std::string &message, const std::exception *cause) throw ()`
- `MessageNotReadableException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~MessageNotReadableException () throw ()`

6.557.1 Detailed Description

This exception must be thrown when a CMS client attempts to read a write-only message.

Since

1.3

6.557.2 Constructor & Destructor Documentation

- 6.557.2.1 `cms::MessageNotReadableException::MessageNotReadableException () throw ()`
- 6.557.2.2 `cms::MessageNotReadableException::MessageNotReadableException (const MessageNotReadableException & ex) throw ()`
- 6.557.2.3 `cms::MessageNotReadableException::MessageNotReadableException (const std::string & message, const std::exception * cause) throw ()`
- 6.557.2.4 `cms::MessageNotReadableException::MessageNotReadableException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.557.2.5 `virtual cms::MessageNotReadableException::~MessageNotReadableException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageNotReadableException.h`

6.558 cms::MessageNotWriteableException Class Reference

This exception must be thrown when a CMS client attempts to write to a read-only message.

```
#include <src/main/cms/MessageNotWriteableException.h>
```

Inheritance diagram for cms::MessageNotWriteableException:

Public Member Functions

- `MessageNotWriteableException () throw ()`
- `MessageNotWriteableException (const MessageNotWriteableException &ex) throw ()`
- `MessageNotWriteableException (const std::string &message, const std::exception *cause) throw ()`
- `MessageNotWriteableException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~MessageNotWriteableException () throw ()`

6.558.1 Detailed Description

This exception must be thrown when a CMS client attempts to write to a read-only message.

Since

1.3

6.558.2 Constructor & Destructor Documentation

- 6.558.2.1 `cms::MessageNotWriteableException::MessageNotWriteableException () throw ()`
- 6.558.2.2 `cms::MessageNotWriteableException::MessageNotWriteableException (const MessageNotWriteableException & ex) throw ()`
- 6.558.2.3 `cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & message, const std::exception * cause) throw ()`
- 6.558.2.4 `cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.558.2.5 `virtual cms::MessageNotWriteableException::~~MessageNotWriteableException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageNotWriteableException.h`

6.559 cms::MessageProducer Class Reference

A client uses a `MessageProducer` (p. 2550) object to send messages to a **Destination** (p. 1610).

```
#include <src/main/cms/MessageProducer.h>
```

Inheritance diagram for `cms::MessageProducer`:

Public Member Functions

- `virtual ~MessageProducer ()`
- `virtual void send (Message *message)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)`
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- `virtual void send (Message *message, int deliveryMode, int priority, long long timeToLive)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)`
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- `virtual void send (const Destination *destination, Message *message)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const **Destination** *destination, **Message** *message, int deliveryMode, int priority, long long timeToLive)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **setDeliveryMode** (int mode)=0 throw (CMSEException)

Sets the delivery mode for this Producer.

- virtual int **getDeliveryMode** () const =0 throw (CMSEException)

Gets the delivery mode for this Producer.

- virtual void **setDisableMessageID** (bool value)=0 throw (CMSEException)

*Sets if **Message** (p. 2375) Ids are disabled for this Producer.*

- virtual bool **getDisableMessageID** () const =0 throw (CMSEException)

*Gets if **Message** (p. 2375) Ids are disabled for this Producer.*

- virtual void **setDisableMessageTimeStamp** (bool value)=0 throw (CMSEException)

*Sets if **Message** (p. 2375) Time Stamps are disabled for this Producer.*

- virtual bool **getDisableMessageTimeStamp** () const =0 throw (CMSEException)

*Gets if **Message** (p. 2375) Time Stamps are disabled for this Producer.*

- virtual void **setPriority** (int priority)=0 throw (CMSEException)

Sets the Priority that this Producers sends messages at.

- virtual int **getPriority** () const =0 throw (CMSEException)

Gets the Priority level that this producer sends messages at.

- virtual void **setTimeToLive** (long long time)=0 throw (CMSEException)

Sets the Time to Live that this Producers sends messages with.

- virtual long long **getTimeToLive** () const =0 throw (CMSEException)

Gets the Time to Live that this producer sends messages with.

6.559.1 Detailed Description

A client uses a **MessageProducer** (p. 2550) object to send messages to a **Destination** (p. 1610). A **MessageProducer** (p. 2550) object is created by passing a **Destination** (p. 1610) object to a message-producer creation method supplied by a session.

A client also has the option of creating a message producer without supplying a destination. In this case, a **Destination** (p. 1610) must be provided with every send operation. A typical use for this kind of message producer is to send replies to requests using the request's CMSReplyTo destination.

A client can specify a default delivery mode, priority, and time to live for messages sent by a message producer. It can also specify the delivery mode, priority, and time to live for an individual message.

A client can specify a time-to-live value in milliseconds for each message it sends. This value defines a message expiration time that is the sum of the message's time-to-live and the GMT when it is sent (for transacted sends, this is the time the client sends the message, not the time the transaction is committed).

Since

1.0

6.559.2 Constructor & Destructor Documentation

6.559.2.1 `virtual cms::MessageProducer::~MessageProducer () [inline, virtual]`

6.559.3 Member Function Documentation

6.559.3.1 `virtual int cms::MessageProducer::getDeliveryMode () const throw (CMSEException) [pure virtual]`

Gets the delivery mode for this Producer.

Returns

The **DeliveryMode** (p. 1609)

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 998), and `activemq::core::ActiveMQProducer` (p. 425).

6.559.3.2 `virtual bool cms::MessageProducer::getDisableMessageID () const throw (CMSEException) [pure virtual]`

Gets if **Message** (p. 2375) Ids are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 998), and `activemq::core::ActiveMQProducer` (p. 425).

6.559.3.3 `virtual bool cms::MessageProducer::getDisableMessageTimeStamp ()
const throw (CMSEException) [pure virtual]`

Gets if **Message** (p. 2375) Time Stamps are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 999), and **activemq::core::ActiveMQProducer** (p. 426).

6.559.3.4 `virtual int cms::MessageProducer::getPriority () const throw (CMSEException) [pure virtual]`

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 999), and **activemq::core::ActiveMQProducer** (p. 426).

6.559.3.5 `virtual long long cms::MessageProducer::getTimeToLive () const
throw (CMSEException) [pure virtual]`

Gets the Time to Live that this producer sends messages with.

Returns

Time to live value in milliseconds

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 999), and **activemq::core::ActiveMQProducer** (p. 427).

6.559.3.6 `virtual void cms::MessageProducer::send (Message * message,
int deliveryMode, int priority, long long timeToLive)
throw (cms::CMSEException, cms::MessageFormatException,
cms::InvalidDestinationException, cms::UnsupportedOperationException) [pure virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSEException (p. 1074) - if an internal error occurs while sending the message.

MessageFormatException (p. 2493) - if an Invalid **Message** (p. 2375) is given.

InvalidDestinationException (p. 1993) - if a client uses this method with a **MessageProducer** (p. 2550) with an invalid destination.

UnsupportedOperationException (p. 3659) - if a client uses this method with a **MessageProducer** (p. 2550) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1000), and **activemq::core::ActiveMQProducer** (p. 428).

6.559.3.7 virtual void cms::MessageProducer::send (const Destination * *destination*, Message * *message*, int *deliveryMode*, int *priority*, long long *timeToLive*) throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters

destination The destination on which to send the message

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSEException (p. 1074) - if an internal error occurs while sending the message.

MessageFormatException (p. 2493) - if an Invalid **Message** (p. 2375) is given.

InvalidDestinationException (p. 1993) - if a client uses this method with a **MessageProducer** (p. 2550) with an invalid destination.

UnsupportedOperationException (p. 3659) - if a client uses this method with a **MessageProducer** (p. 2550) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1001), and **activemq::core::ActiveMQProducer** (p. 427).

6.559.3.8 virtual void cms::MessageProducer::send (Message * *message*) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [pure virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

message The message to be sent.

Exceptions

CMSException (p. 1074) - if an internal error occurs while sending the message.

MessageFormatException (p. 2493) - if an Invalid **Message** (p. 2375) is given.

InvalidDestinationException (p. 1993) - if a client uses this method with a **MessageProducer** (p. 2550) with an invalid destination.

UnsupportedOperationException (p. 3659) - if a client uses this method with a **MessageProducer** (p. 2550) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1001), and **activemq::core::ActiveMQProducer** (p. 428).

6.559.3.9 virtual void cms::MessageProducer::send (const Destination * *destination*, Message * *message*) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

destination The destination on which to send the message

message the message to be sent.

Exceptions

CMSException (p. 1074) - if an internal error occurs while sending the message.

MessageFormatException (p. 2493) - if an Invalid **Message** (p. 2375) is given.

InvalidDestinationException (p. 1993) - if a client uses this method with a **MessageProducer** (p. 2550) with an invalid destination.

UnsupportedOperationException (p. 3659) - if a client uses this method with a **MessageProducer** (p. 2550) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1000), and **activemq::core::ActiveMQProducer** (p. 429).

6.559.3.10 `virtual void cms::MessageProducer::setDeliveryMode (int mode)
throw (CMSEException)` [pure virtual]

Sets the delivery mode for this Producer.

Parameters

mode The **DeliveryMode** (p. 1609)

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1002), and **activemq::core::ActiveMQProducer** (p. 429).

6.559.3.11 `virtual void cms::MessageProducer::setDisableMessageID (bool value)
throw (CMSEException)` [pure virtual]

Sets if **Message** (p. 2375) Ids are disabled for this Producer.

Parameters

value boolean indicating enable / disable (true / false)

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1002), and **activemq::core::ActiveMQProducer** (p. 430).

6.559.3.12 `virtual void cms::MessageProducer::setDisableMessageTimeStamp (bool value)
throw (CMSEException)` [pure virtual]

Sets if **Message** (p. 2375) Time Stamps are disabled for this Producer.

Parameters

value - boolean indicating enable / disable (true / false)

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1002), and **activemq::core::ActiveMQProducer** (p. 430).

6.559.3.13 `virtual void cms::MessageProducer::setPriority (int priority)
throw (CMSEException)` [pure virtual]

Sets the Priority that this Producers sends messages at.

Parameters

priority int value for Priority level

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 1002), and `activemq::core::ActiveMQProducer` (p. 430).

6.559.3.14 `virtual void cms::MessageProducer::setTimeToLive (long long time) throw (CMSEException)` [pure virtual]

Sets the Time to Live that this Producers sends messages with.

This value will be used if the time to live is not specified via the send method.

Parameters

time default time to live value in milliseconds

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 1003), and `activemq::core::ActiveMQProducer` (p. 430).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageProducer.h`

6.560 activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

```
#include <src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
```

Public Member Functions

- **MessagePropertyInterceptor** (`commands::Message *message`, `util::PrimitiveMap *properties`) throw (`decaf::lang::exceptions::NullPointerException`)

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

- `virtual ~MessagePropertyInterceptor ()`
- `virtual bool getBooleanProperty (const std::string &name) const`

Gets a boolean property.

- virtual unsigned char **getByteProperty** (const std::string &name) const
Gets a byte property.
- virtual double **getDoubleProperty** (const std::string &name) const
Gets a double property.
- virtual float **getFloatProperty** (const std::string &name) const
Gets a float property.
- virtual int **getIntProperty** (const std::string &name) const
Gets a int property.
- virtual long long **getLongProperty** (const std::string &name) const
Gets a long property.
- virtual short **getShortProperty** (const std::string &name) const
Gets a short property.
- virtual std::string **getStringProperty** (const std::string &name) const
Gets a string property.
- virtual void **setBooleanProperty** (const std::string &name, bool value)
Sets a boolean property.
- virtual void **setByteProperty** (const std::string &name, unsigned char value)
Sets a byte property.
- virtual void **setDoubleProperty** (const std::string &name, double value)
Sets a double property.
- virtual void **setFloatProperty** (const std::string &name, float value)
Sets a float property.
- virtual void **setIntProperty** (const std::string &name, int value)
Sets a int property.
- virtual void **setLongProperty** (const std::string &name, long long value)
Sets a long property.
- virtual void **setShortProperty** (const std::string &name, short value)
Sets a short property.
- virtual void **setStringProperty** (const std::string &name, const std::string &value)
Sets a string property.

6.560.1 Detailed Description

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties. Currently the only properties that are intercepted and handled are:

Name | Conversion Supported ----- JMSXDeliveryCount |
Int, Long, String JMSXGroupID | String JMSXGroupSeq | Int, Long, String

6.560.2 Constructor & Destructor Documentation

6.560.2.1 `activemq::wireformat::openwire::utils::MessagePropertyInterceptor::MessagePropertyInterceptor (commands::Message * message, util::PrimitiveMap * properties)`
`throw (decaf::lang::exceptions::NullPointerException)`

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

Parameters

message - The Message to store reserved property data in
properties - The PrimitiveMap to store the rest of the properties in.

Exceptions

NullPointerException if either param is NULL

6.560.2.2 `virtual`
`activemq::wireformat::openwire::utils::MessagePropertyInterceptor::~MessagePropertyInterceptor () [virtual]`

6.560.3 Member Function Documentation

6.560.3.1 `virtual bool` `activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getBooleanProperty (const std::string & name) const [virtual]`

Gets a boolean property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.560.3.2 `virtual unsigned char` `activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getByteProperty (const std::string & name) const [virtual]`

Gets a byte property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.560.3.3 `virtual double activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getDoubleProperty (const std::string & name) const [virtual]`

Gets a double property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.560.3.4 `virtual float activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getFloatProperty (const std::string & name) const [virtual]`

Gets a float property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.560.3.5 `virtual int activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getIntProperty (const std::string & name) const [virtual]`

Gets a int property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.560.3.6 `virtual long long ac-`
 `tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::getLongProperty`
 `(const std::string & name) const [virtual]`

Gets a long property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.560.3.7 `virtual short ac-`
 `tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::getShortProperty`
 `(const std::string & name) const [virtual]`

Gets a short property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.560.3.8 `virtual std::string ac-`
 `tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::getStringProperty`
 `(const std::string & name) const [virtual]`

Gets a string property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.560.3.9 `virtual void ac-`
 `tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::setBooleanProperty`
 `(const std::string & name, bool value) [virtual]`

Sets a boolean property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.560.3.10 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setByteProperty (const std::string & name, unsigned char value) [virtual]`

Sets a byte property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.560.3.11 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setDoubleProperty (const std::string & name, double value) [virtual]`

Sets a double property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.560.3.12 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setFloatProperty (const std::string & name, float value) [virtual]`

Sets a float property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.560.3.13 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setIntProperty (const std::string & name, int value) [virtual]`

Sets a int property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.560.3.14 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setLongProperty (const std::string & name, long long value) [virtual]`

Sets a long property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.560.3.15 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setShortProperty (const std::string & name, short value) [virtual]`

Sets a short property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.560.3.16 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setStringProperty (const std::string & name, const std::string & value) [virtual]`

Sets a string property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h`

6.561 activemq::commands::MessagePull Class Reference

```
#include <src/main/activemq/commands/MessagePull.h>
```

Inheritance diagram for `activemq::commands::MessagePull`:

Public Member Functions

- `MessagePull ()`
- `virtual ~MessagePull ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual MessagePull * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long timeout)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &correlationId)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEPULL** = 20

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **timeout**
- std::string **correlationId**
- **Pointer**< **MessageId** > **messageId**

6.561.1 Constructor & Destructor Documentation

6.561.1.1 `activemq::commands::MessagePull::MessagePull ()`

6.561.1.2 `virtual activemq::commands::MessagePull::~~MessagePull () [virtual]`

6.561.2 Member Function Documentation

6.561.2.1 `virtual MessagePull* activemq::commands::MessagePull::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.561.2.2 `virtual void activemq::commands::MessagePull::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.561.2.3 `virtual bool activemq::commands::MessagePull::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

- 6.561.2.4** `virtual const Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId () const [virtual]`
- 6.561.2.5** `virtual Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId () [virtual]`
- 6.561.2.6** `virtual const std::string& activemq::commands::MessagePull::getCorrelationId () const [virtual]`
- 6.561.2.7** `virtual std::string& activemq::commands::MessagePull::getCorrelationId () [virtual]`
- 6.561.2.8** `virtual unsigned char activemq::commands::MessagePull::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.561.2.9 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination () const [virtual]
- 6.561.2.10 virtual Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination () [virtual]
- 6.561.2.11 virtual const Pointer<MessageId>& activemq::commands::MessagePull::getMessageId () const [virtual]
- 6.561.2.12 virtual Pointer<MessageId>& activemq::commands::MessagePull::getMessageId () [virtual]
- 6.561.2.13 virtual long long activemq::commands::MessagePull::getTimeout () const [virtual]
- 6.561.2.14 virtual void activemq::commands::MessagePull::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]
- 6.561.2.15 virtual void activemq::commands::MessagePull::setCorrelationId (const std::string & *correlationId*) [virtual]
- 6.561.2.16 virtual void activemq::commands::MessagePull::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.561.2.17 virtual void activemq::commands::MessagePull::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]
- 6.561.2.18 virtual void activemq::commands::MessagePull::setTimeout (long long *timeout*) [virtual]
- 6.561.2.19 virtual std::string activemq::commands::MessagePull::toString () const [virtual]

Returns a string containing the information for this **DataSet** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

- 6.561.2.20 virtual Pointer<Command> activemq::commands::MessagePull::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.561.3 Field Documentation

- 6.561.3.1** **Pointer<ConsumerId> activemq::commands::MessagePull::consumerId**
[protected]
- 6.561.3.2** **std::string activemq::commands::MessagePull::correlationId** [protected]
- 6.561.3.3** **Pointer<ActiveMQDestination> activemq::commands::MessagePull::destination** [protected]
- 6.561.3.4** **const unsigned char activemq::commands::MessagePull::ID _-MESSAGEPULL = 20** [static]
- 6.561.3.5** **Pointer<MessageId> activemq::commands::MessagePull::messageId**
[protected]
- 6.561.3.6** **long long activemq::commands::MessagePull::timeout** [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessagePull.h`

6.562 **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2568).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller**:

Public Member Functions

- **MessagePullMarshaller ()**
- **virtual ~MessagePullMarshaller ()**
- **virtual commands::DataStructure * createObject () const**
Creates a new instance of this marshalable type.
- **virtual unsigned char getDataStructureType () const**
Get the Data Structure Type that identifies this Marshaler.
- **virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)**

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.562.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2568). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.562.2 Constructor & Destructor Documentation

6.562.2.1 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::MessagePullMarshaller () [inline]

6.562.2.2 virtual activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::~~MessagePullMarshaller () [inline, virtual]

6.562.3 Member Function Documentation

6.562.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.562.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.562.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.562.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

6.562.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 738).

6.562.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 739).

6.562.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h

6.563 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2572).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller**:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.563.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2572). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.563.2 Constructor & Destructor Documentation

6.563.2.1 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::MessagePullMarshaller () [inline]

6.563.2.2 virtual activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::~~MessagePullMarshaller () [inline, virtual]

6.563.3 Member Function Documentation

6.563.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.563.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.563.3.3 virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.563.3.4 virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

6.563.3.5 virtual int activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```
6.563.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.563.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h`

6.564 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2576).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller**:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.564.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2576). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.564.2 Constructor & Destructor Documentation

6.564.2.1 `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::MessagePullMarshaller ()` [inline]

6.564.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::~~MessagePullMarshaller ()` [inline, virtual]

6.564.3 Member Function Documentation

6.564.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::createObject () const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.564.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::getDataStructureType () const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.564.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.564.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.564.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

```
6.564.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.564.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h`

6.565 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2580).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller`:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.565.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2580). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.565.2 Constructor & Destructor Documentation

6.565.2.1 `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::MessagePullMarshaller ()` [inline]

6.565.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::~~MessagePullMarshaller ()` [inline, virtual]

6.565.3 Member Function Documentation

6.565.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::createObject () const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.565.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::getDataStructureType () const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.565.3.3 virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.565.3.4 virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

6.565.3.5 virtual int activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 711).

```
6.565.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 712).

```
6.565.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h`

6.566 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2584).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller**:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.566.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2584). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.566.2 Constructor & Destructor Documentation

6.566.2.1 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::MessagePullMarshaller () [inline]

6.566.2.2 virtual
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::~~MessagePullMarshaller () [inline, virtual]

6.566.3 Member Function Documentation

6.566.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.566.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.566.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.566.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.566.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 718).

```
6.566.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 719).

```
6.566.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h`

6.567 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2588).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller**:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.567.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2588). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.567.2 Constructor & Destructor Documentation

6.567.2.1 **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::MessagePullMarshaller () [inline]**

6.567.2.2 **virtual**
activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::~~MessagePullMarshaller () [inline, virtual]

6.567.3 Member Function Documentation

6.567.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::createObject () const [virtual]**

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.567.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::getDataStructureType () const [virtual]**

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.567.3.3 virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.567.3.4 virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

6.567.3.5 virtual int activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

```
6.567.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

```
6.567.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h`

6.568 activemq::transport::mock::MockTransport Class Reference

The **MockTransport** (p. 2592) defines a base level **Transport** (p. 3629) class that is intended to be used in place of an a regular protocol **Transport** (p. 3629) such as TCP.

```
#include <src/main/activemq/transport/mock/MockTransport.h>
```

Inheritance diagram for `activemq::transport::mock::MockTransport`:

Public Member Functions

- **MockTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat, const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual **~MockTransport** ()
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)
*Sets the **ResponseBuilder** (p. 3079) that this class uses to create Responses to Commands sent.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual void **setOutgoingListener** (**TransportListener** *listener)
Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat **AMQCPP_UNUSED**)
Sets the WireFormat instance to use.

- **Pointer< wireformat::WireFormat > getWireFormat ()** const
Gets the currently set WireFormat.
- virtual void **setTransportListener** (**TransportListener** *listener)
Sets the observer of asynchronous exceptions from this transport.
- virtual **TransportListener** * **getTransportListener** () const
Gets the observer of asynchronous exceptions from this transport.
- virtual void **fireCommand** (const **Pointer< Command >** &command)
Fires a Command back through this transport to its registered CommandListener if there is one.
- virtual void **fireException** (const **exceptions::ActiveMQException** &ex)
Fires a Exception back through this transport to its registered ExceptionListener if there is one.
- virtual void **start** () throw (**decaf::io::IOException**)
*Starts the **Transport** (p. 3629), the send methods of a **Transport** (p. 3629) will throw an exception if used before the **Transport** (p. 3629) is started.*
- virtual void **stop** () throw (**decaf::io::IOException**)
*Stops the **Transport** (p. 3629).*
- virtual void **close** () throw (**decaf::io::IOException**)
Closes this object and deallocates the appropriate resources.
- virtual **Transport** * **narrow** (const std::type_info &typeId)
*Narrows down a Chain of Transports to a specific **Transport** (p. 3629) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
*Is the **Transport** (p. 3629) Connected to its Broker.*
- virtual bool **isClosed** () const
*Has the **Transport** (p. 3629) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri AMQCPP_UNUSED) throw (**decaf::io::IOException**)
reconnect to another location
- bool **isFailOnSendMessage** () const
- void **setFailOnSendMessage** (bool value)
- int **getNumSentMessageBeforeFail** () const
- void **setNumSentMessageBeforeFail** (int value)
- int **getNumSentMessages** () const

- void **setNumSentMessages** (int value)
- bool **isFailOnReceiveMessage** () const
- void **setFailOnReceiveMessage** (bool value)
- int **getNumReceivedMessageBeforeFail** () const
- void **setNumReceivedMessageBeforeFail** (int value)
- int **getNumReceivedMessages** () const
- void **setNumReceivedMessages** (int value)
- bool **isFailOnKeepAliveSends** () const
- void **setFailOnKeepAliveSends** (bool value)
- int **getNumSentKeepAlivesBeforeFail** () const
- void **setNumSentKeepAlivesBeforeFail** (int value)
- int **getNumSentKeepAlives** () const
- void **setNumSentKeepAlives** (int value)
- bool **isFailOnStart** () const
- void **setFailOnStart** (bool value)
- bool **isFailOnStop** () const
- void **setFailOnStop** (bool value)
- bool **isFailOnClose** () const
- void **setFailOnClose** (bool value)

Static Public Member Functions

- static **MockTransport * getInstance** ()

6.568.1 Detailed Description

The **MockTransport** (p. 2592) defines a base level **Transport** (p. 3629) class that is intended to be used in place of an a regular protocol **Transport** (p. 3629) such as TCP. This **Transport** (p. 3629) assumes that it is the base **Transport** (p. 3629) in the Transports stack, and destroys any Transports that are passed to it in its constructor.

This **Transport** (p. 3629) defines an Interface **ResponseBuilder** (p. 3079) which must be implemented by any protocol for which the **Transport** (p. 3629) is used to Emulate. The **Transport** (p. 3629) hands off all outbound commands to the **ResponseBuilder** (p. 3079) for processing, it is up to the builder to create appropriate responses and schedule any asynchronous messages that might result from a message sent to the Broker.

6.568.2 Constructor & Destructor Documentation

6.568.2.1 `activemq::transport::mock::MockTransport::MockTransport (const Pointer< wireformat::WireFormat > & wireFormat, const Pointer< ResponseBuilder > & responseBuilder)`

6.568.2.2 `virtual activemq::transport::mock::MockTransport::~MockTransport () [inline, virtual]`

6.568.3 Member Function Documentation

6.568.3.1 `virtual void activemq::transport::mock::MockTransport::close () throw (decaf::io::IOException) [virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException if an error occurs while closing.

Implements **decaf::io::Closeable** (p. 1066).

6.568.3.2 `virtual void activemq::transport::mock::MockTransport::fireCommand (const Pointer< Command > & command) [inline, virtual]`

Fires a Command back through this transport to its registered CommandListener if there is one.

Parameters

command - Command to send to the Listener.

6.568.3.3 `virtual void activemq::transport::mock::MockTransport::fireException (const exceptions::ActiveMQException & ex) [inline, virtual]`

Fires a Exception back through this transport to its registered ExceptionListener if there is one.

Parameters

ex The Exception that will be passed on the the **Transport** (p. 3629) listener.

- 6.568.3.4** `static MockTransport* activemq::transport::mock::MockTransport::getInstance () [inline, static]`
- 6.568.3.5** `int activemq::transport::mock::MockTransport::getNumReceivedMessageBeforeFail () const [inline]`
- 6.568.3.6** `int activemq::transport::mock::MockTransport::getNumReceivedMessages () const [inline]`
- 6.568.3.7** `int activemq::transport::mock::MockTransport::getNumSentKeepAlives () const [inline]`
- 6.568.3.8** `int activemq::transport::mock::MockTransport::getNumSentKeepAlivesBeforeFail () const [inline]`
- 6.568.3.9** `int activemq::transport::mock::MockTransport::getNumSentMessageBeforeFail () const [inline]`
- 6.568.3.10** `int activemq::transport::mock::MockTransport::getNumSentMessages () const [inline]`
- 6.568.3.11** `virtual std::string activemq::transport::mock::MockTransport::getRemoteAddress () const [inline, virtual]`

Returns

the remote address for this connection

Implements `activemq::transport::Transport` (p. 3630).

- 6.568.3.12** `virtual TransportListener* activemq::transport::mock::MockTransport::getTransportListener () const [inline, virtual]`

Gets the observer of asynchronous exceptions from this transport.

Returns

The listener of transport events.

Implements `activemq::transport::Transport` (p. 3631).

- 6.568.3.13** `Pointer<wireformat::WireFormat> activemq::transport::mock::MockTransport::getWireFormat () const [inline]`

Gets the currently set WireFormat.

Returns

the current WireFormat object.

6.568.3.14 `virtual bool activemq::transport::mock::MockTransport::isClosed ()
const [inline, virtual]`

Has the **Transport** (p. 3629) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3629)

Implements **activemq::transport::Transport** (p. 3631).

6.568.3.15 `virtual bool activemq::transport::mock::MockTransport::isConnected ()
const [inline, virtual]`

Is the **Transport** (p. 3629) Connected to its Broker.

Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3631).

6.568.3.16 `bool activemq::transport::mock::MockTransport::isFailOnClose ()
const [inline]`

6.568.3.17 `bool activemq::transport::mock::MockTransport::isFailOnKeepAliveSends ()
const [inline]`

6.568.3.18 `bool activemq::transport::mock::MockTransport::isFailOnReceiveMessage ()
const [inline]`

6.568.3.19 `bool activemq::transport::mock::MockTransport::isFailOnSendMessage ()
const [inline]`

6.568.3.20 `bool activemq::transport::mock::MockTransport::isFailOnStart ()
const [inline]`

6.568.3.21 `bool activemq::transport::mock::MockTransport::isFailOnStop ()
const [inline]`

6.568.3.22 `virtual bool activemq::transport::mock::MockTransport::isFaultTolerant ()
const [inline, virtual]`

Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3629) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3631).

6.568.3.23 `virtual Transport* activemq::transport::mock::MockTransport::narrow (const std::type_info & typeId) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3629) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

typeId - The type_info of the Object we are searching for.

Returns

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3631).

6.568.3.24 `virtual void activemq::transport::mock::MockTransport::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3632).

6.568.3.25 `virtual void activemq::transport::mock::MockTransport::reconnect (const decaf::net::URI &uri AMQCPP_UNUSED) throw (decaf::io::IOException) [inline, virtual]`

reconnect to another location

Parameters

uri

Exceptions

IOException on failure of if not supported

6.568.3.26 `virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

command - The command to be sent.

timeout - The time to wait for this response.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements `activemq::transport::Transport` (p. 3633).

6.568.3.27 `virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

command the command to be sent.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements `activemq::transport::Transport` (p. 3632).

- 6.568.3.28 `void activemq::transport::mock::MockTransport::setFailOnClose (bool value) [inline]`
- 6.568.3.29 `void activemq::transport::mock::MockTransport::setFailOnKeepAliveSends (bool value) [inline]`
- 6.568.3.30 `void activemq::transport::mock::MockTransport::setFailOnReceiveMessage (bool value) [inline]`
- 6.568.3.31 `void activemq::transport::mock::MockTransport::setFailOnSendMessage (bool value) [inline]`
- 6.568.3.32 `void activemq::transport::mock::MockTransport::setFailOnStart (bool value) [inline]`
- 6.568.3.33 `void activemq::transport::mock::MockTransport::setFailOnStop (bool value) [inline]`
- 6.568.3.34 `void activemq::transport::mock::MockTransport::setNumReceivedMessageBeforeFail (int value) [inline]`
- 6.568.3.35 `void activemq::transport::mock::MockTransport::setNumReceivedMessages (int value) [inline]`
- 6.568.3.36 `void activemq::transport::mock::MockTransport::setNumSentKeepAlives (int value) [inline]`
- 6.568.3.37 `void activemq::transport::mock::MockTransport::setNumSentKeepAlivesBeforeFail (int value) [inline]`
- 6.568.3.38 `void activemq::transport::mock::MockTransport::setNumSentMessageBeforeFail (int value) [inline]`
- 6.568.3.39 `void activemq::transport::mock::MockTransport::setNumSentMessages (int value) [inline]`
- 6.568.3.40 `virtual void activemq::transport::mock::MockTransport::setOutgoingListener (TransportListener * listener) [inline, virtual]`

Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.

Parameters

listener - The CommandListener to notify for each message

6.568.3.41 `void activemq::transport::mock::MockTransport::setResponseBuilder (const Pointer< ResponseBuilder > & responseBuilder) [inline]`

Sets the **ResponseBuilder** (p. 3079) that this class uses to create Responses to Commands sent. These are either real Response Objects, or Commands that would have been sent Asynchronously be the Broker.

Parameters

responseBuilder - The **ResponseBuilder** (p. 3079) to use from now on.

6.568.3.42 `virtual void activemq::transport::mock::MockTransport::setTransportListener (TransportListener * listener) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

Parameters

listener the listener of transport events.

Implements **activemq::transport::Transport** (p. 3633).

6.568.3.43 `virtual void activemq::transport::mock::MockTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat AMQCPP_UNUSED) [inline, virtual]`

Sets the WireFormat instance to use.

Parameters

wireFormat WireFormat the object used to encode / decode commands.

6.568.3.44 `virtual void activemq::transport::mock::MockTransport::start () throw (decaf::io::IOException) [virtual]`

Starts the **Transport** (p. 3629), the send methods of a **Transport** (p. 3629) will throw an exception if used before the **Transport** (p. 3629) is started.

Exceptions

IOException if and error occurs while starting the **Transport** (p. 3629).

Implements **activemq::transport::Transport** (p. 3634).

6.568.3.45 `virtual void activemq::transport::mock::MockTransport::stop () throw (decaf::io::IOException) [virtual]`

Stops the **Transport** (p. 3629).

Exceptions

IOException if an error occurs while stopping the transport.

Implements **activemq::transport::Transport** (p. 3634).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/MockTransport.h`

6.569 **activemq::transport::mock::MockTransportFactory** Class Reference

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

```
#include <src/main/activemq/transport/mock/MockTransportFactory.h>
```

Inheritance diagram for **activemq::transport::mock::MockTransportFactory**:

Public Member Functions

- virtual **~MockTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)
*Creates a fully configured **Transport** (p. 3629) instance which could be a chain of filters and transports.*
- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)
*Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **Pointer< wireformat::WireFormat >** &wireFormat, const **decaf::util::Properties** &properties) throw (exceptions::ActiveMQException)
*Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.*

6.569.1 Detailed Description

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

6.569.2 Constructor & Destructor Documentation

6.569.2.1 virtual
activemq::transport::mock::MockTransportFactory::~MockTransportFactory
() [inline, virtual]

6.569.3 Member Function Documentation

6.569.3.1 virtual Pointer<Transport> ac-
tivemq::transport::mock::MockTransportFactory::create
(const decaf::net::URI & *location*) throw (
exceptions::ActiveMQException) [virtual]

Creates a fully configured **Transport** (p.3629) instance which could be a chain of filters and transports.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p.3635).

6.569.3.2 virtual Pointer<Transport> ac-
tivemq::transport::mock::MockTransportFactory::createComposite
(const decaf::net::URI & *location*) throw (
exceptions::ActiveMQException) [virtual]

Creates a slimed down **Transport** (p.3629) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p.3635).

6.569.3.3 virtual Pointer<Transport> ac-
tivemq::transport::mock::MockTransportFactory::doCreateComposite
(const decaf::net::URI & *location*, const Pointer<
wireformat::WireFormat> & *wireFormat*, const decaf::util::Properties
& *properties*) throw (exceptions::ActiveMQException) [protected,
virtual]

Creates a slimed down **Transport** (p.3629) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to.

wireFormat - the assigned WireFormat for the new **Transport** (p. 3629).

properties - Properties to apply to the transport.

Returns

Pointer to a new **Transport** (p. 3629) instance.

Exceptions

ActiveMQException if an error occurs

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**MockTransportFactory.h**

6.570 decaf::util::concurrent::Mutex Class Reference

Mutex (p. 2604) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

#include <src/main/decaf/util/concurrent/Mutex.h>

Inheritance diagram for decaf::util::concurrent::Mutex:

Public Member Functions

- **Mutex** ()
- virtual ~**Mutex** ()
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
*Attempts to **Lock** (p. 2228) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.570.1 Detailed Description

Mutex (p.2604) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

Since

1.0

6.570.2 Constructor & Destructor Documentation

6.570.2.1 decaf::util::concurrent::Mutex::Mutex ()

6.570.2.2 virtual decaf::util::concurrent::Mutex::~~Mutex () [virtual]

6.570.3 Member Function Documentation

6.570.3.1 virtual void decaf::util::concurrent::Mutex::lock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements decaf::util::concurrent::Synchronizable (p.3463).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::lock(), decaf::util::StlMap< std::string, cms::Topic * >::lock(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::lock(), and decaf::util::AbstractCollection< cms::Connection * >::lock().

6.570.3.2 `virtual void decaf::util::concurrent::Mutex::notify ()
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3464).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notify()`, `decaf::util::StlMap< std::string, cms::Topic * >::notify()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notify()`, and `decaf::util::AbstractCollection< cms::Connection * >::notify()`.

6.570.3.3 `virtual void decaf::util::concurrent::Mutex::notifyAll
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3465).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notifyAll()`, `decaf::util::StlMap< std::string, cms::Topic * >::notifyAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notifyAll()`, and `decaf::util::AbstractCollection< cms::Connection * >::notifyAll()`.

6.570.3.4 `virtual bool decaf::util::concurrent::Mutex::tryLock () throw (decaf::lang::exceptions::RuntimeException) [virtual]`

Attempts to **Lock** (p. 2228) the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3466).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::tryLock()`, `decaf::util::StlMap< std::string, cms::Topic * >::tryLock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::tryLock()`, and `decaf::util::AbstractCollection< cms::Connection * >::tryLock()`.

6.570.3.5 virtual void decaf::util::concurrent::Mutex::unlock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3467).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::unlock()`, `decaf::util::StlMap< std::string, cms::Topic * >::unlock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::unlock()`, and `decaf::util::AbstractCollection< cms::Connection * >::unlock()`.

6.570.3.6 virtual void decaf::util::concurrent::Mutex::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3468).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::wait()`, `decaf::util::StlMap< std::string, cms::Topic * >::wait()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::wait()`, and `decaf::util::AbstractCollection< cms::Connection * >::wait()`.

6.570.3.7 `virtual void decaf::util::concurrent::Mutex::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3470).

6.570.3.8 `virtual void decaf::util::concurrent::Mutex::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3471).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Mutex.h**

6.571 decaf::util::concurrent::MutexHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/MutexHandle.h>
```

Public Member Functions

- **MutexHandle** ()
- **~MutexHandle** ()
- **MutexHandle** ()
- **~MutexHandle** ()

Data Fields

- pthread_mutex_t **mutex**
- volatile long long **lock_owner**
- volatile long long **lock_count**
- CRITICAL_SECTION **mutex**

6.571.1 Constructor & Destructor Documentation

6.571.1.1 decaf::util::concurrent::MutexHandle::MutexHandle () [inline]

6.571.1.2 decaf::util::concurrent::MutexHandle::~~MutexHandle () [inline]

6.571.1.3 decaf::util::concurrent::MutexHandle::MutexHandle () [inline]

6.571.1.4 decaf::util::concurrent::MutexHandle::~~MutexHandle () [inline]

6.571.2 Field Documentation

6.571.2.1 volatile long long decaf::util::concurrent::MutexHandle::lock_count

6.571.2.2 volatile long long decaf::util::concurrent::MutexHandle::lock_owner

6.571.2.3 CRITICAL_SECTION decaf::util::concurrent::MutexHandle::mutex

6.571.2.4 pthread_mutex_t decaf::util::concurrent::MutexHandle::mutex

The documentation for this class was generated from the following files:

- src/main/decaf/internal/util/concurrent/unix/**MutexHandle.h**
- src/main/decaf/internal/util/concurrent/windows/**MutexHandle.h**

6.572 decaf::internal::util::concurrent::MutexImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/MutexImpl.h>
```

Static Public Member Functions

- static **decaf::util::concurrent::MutexHandle * create** ()
Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.
- static void **destroy** (decaf::util::concurrent::MutexHandle *handle)
Destroy a previously create Mutex instance.
- static void **lock** (decaf::util::concurrent::MutexHandle *handle)
Locks the Mutex.
- static bool **trylock** (decaf::util::concurrent::MutexHandle *handle)
Tries to lock the Mutex.
- static void **unlock** (decaf::util::concurrent::MutexHandle *handle)
Unlocks the Mutex allowing other Thread to then acquire the Lock on it.

6.572.1 Member Function Documentation

6.572.1.1 static decaf::util::concurrent::MutexHandle* decaf::internal::util::concurrent::MutexImpl::create () [static]

Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.

Returns

handle to a newly created Mutex.

6.572.1.2 static void decaf::internal::util::concurrent::MutexImpl::destroy (decaf::util::concurrent::MutexHandle * *handle*) [static]

Destroy a previously create Mutex instance.

Parameters

mutex The Mutex instance to be destroyed.

6.572.1.3 static void decaf::internal::util::concurrent::MutexImpl::lock (decaf::util::concurrent::MutexHandle * *handle*) [static]

Locks the Mutex.

If the Mutex is already locked by another thread this method blocks until the Mutex becomes unlocked and this thread acquires the lock.

Parameters

handle the handle to the Mutex to Lock.

6.572.1.4 static bool decaf::internal::util::concurrent::MutexImpl::trylock (decaf::util::concurrent::MutexHandle * *handle*) [static]

Tries to lock the Mutex.

If the Mutex is unlocked this Thread acquires the lock on the Mutex and this method returns true, if the Mutex is already locked then the lock is not acquired and this method returns false.

Parameters

handle the handle to the Mutex to attempt to Lock.

Returns

true if the lock was acquired false otherwise.

6.572.1.5 static void decaf::internal::util::concurrent::MutexImpl::unlock (decaf::util::concurrent::MutexHandle * *handle*) [static]

Unlocks the Mutex allowing other Thread to then acquire the Lock on it.

Parameters

handle the handle to the Mutex to attempt to Lock.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/MutexImpl.h

6.573 decaf::internal::net::Network Class Reference

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

```
#include <src/main/decaf/internal/net/Network.h>
```

Public Member Functions

- virtual ~**Network** ()
- decaf::util::concurrent::Mutex * getRuntimeLock ()
*Gets a pointer to the **Network** (p. 2611) Runtime's Lock object, this can be used by **Network** (p. 2611) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 2611) layer, etc.*
- void addNetworkResource (decaf::internal::util::Resource *value)
*Adds a Resource to the **Network** (p. 2611) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 2611) Runtime are destroyed.*
- template<typename T >
void addAsResource (T *value)

Static Public Member Functions

- static **Network** * **getNetworkRuntime** ()

*Gets the one and only instance of the **Network** (p. 2611) class, if this is called before the **Network** (p. 2611) layer has been initialized or after it has been shutdown then an *IllegalStateException* is thrown.*

- static void **initializeNetworking** ()

Initialize the Networking layer.

- static void **shutdownNetworking** ()

*Shutdown the **Network** (p. 2611) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.*

Protected Member Functions

- **Network** ()

6.573.1 Detailed Description

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

Since

1.0

6.573.2 Constructor & Destructor Documentation

6.573.2.1 `decaf::internal::net::Network::Network ()` [protected]

6.573.2.2 `virtual decaf::internal::net::Network::~~Network ()` [virtual]

6.573.3 Member Function Documentation

6.573.3.1 `template<typename T > void decaf::internal::net::Network::addAsResource (T * value)` [inline]

6.573.3.2 `void decaf::internal::net::Network::addNetworkResource (decaf::internal::util::Resource * value)`

Adds a Resource to the **Network** (p. 2611) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 2611) Runtime are destroyed.

Parameters

value The Resource to add to the **Network** (p. 2611) Runtime.

Exceptions

NullPointerException if the Resource value passed is null.

6.573.3.3 static Network* decaf::internal::net::Network::getNetworkRuntime () [static]

Gets the one and only instance of the **Network** (p. 2611) class, if this is called before the **Network** (p. 2611) layer has been initialized or after it has been shutdown then an `IllegalStateException` is thrown.

Returns

pointer to the **Network** (p. 2611) runtime for the Decaf library.

6.573.3.4 decaf::util::concurrent::Mutex* decaf::internal::net::Network::getRuntimeLock ()

Gets a pointer to the **Network** (p. 2611) Runtime's Lock object, this can be used by **Network** (p. 2611) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 2611) layer, etc.

The pointer returned is owned by the **Network** (p. 2611) runtime and should not be deleted or copied by the caller.

Returns

a pointer to the **Network** (p. 2611) Runtime's single Lock instance.

6.573.3.5 static void decaf::internal::net::Network::initializeNetworking () [static]

Initialize the Networking layer.

6.573.3.6 static void decaf::internal::net::Network::shutdownNetworking () [static]

Shutdown the **Network** (p. 2611) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/Network.h`

6.574 activemq::commands::NetworkBridgeFilter Class Reference

```
#include <src/main/activemq/commands/NetworkBridgeFilter.h>
```

Inheritance diagram for `activemq::commands::NetworkBridgeFilter`:

Public Member Functions

- **NetworkBridgeFilter** ()
- virtual **~NetworkBridgeFilter** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **NetworkBridgeFilter * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual int **getNetworkTTL** () const
- virtual void **setNetworkTTL** (int networkTTL)
- virtual const **Pointer**< **BrokerId** > & **getNetworkBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getNetworkBrokerId** ()
- virtual void **setNetworkBrokerId** (const **Pointer**< **BrokerId** > &networkBrokerId)

Static Public Attributes

- static const unsigned char **ID_NETWORKBRIDGEFILTER** = 91

Protected Attributes

- int **networkTTL**
- **Pointer**< **BrokerId** > **networkBrokerId**

6.574.1 Constructor & Destructor Documentation

6.574.1.1 `activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter ()`

6.574.1.2 `virtual
activemq::commands::NetworkBridgeFilter::~~NetworkBridgeFilter ()
[virtual]`

6.574.2 Member Function Documentation

6.574.2.1 `virtual NetworkBridgeFilter* ac-
tivemq::commands::NetworkBridgeFilter::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.574.2.2 `virtual void ac-
tivemq::commands::NetworkBridgeFilter::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1555).

6.574.2.3 `virtual bool activemq::commands::NetworkBridgeFilter::equals (const
DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1556).

6.574.2.4 `virtual unsigned char ac-
tivemq::commands::NetworkBridgeFilter::getDataStructureType ()
const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1553) type copy.

Implements **activemq::commands::DataSet** (p. 1557).

- 6.574.2.5** `virtual const Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId () const [virtual]`
- 6.574.2.6** `virtual Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId () [virtual]`
- 6.574.2.7** `virtual int activemq::commands::NetworkBridgeFilter::getNetworkTTL () const [virtual]`
- 6.574.2.8** `virtual void activemq::commands::NetworkBridgeFilter::setNetworkBrokerId (const Pointer< BrokerId > & networkBrokerId) [virtual]`
- 6.574.2.9** `virtual void activemq::commands::NetworkBridgeFilter::setNetworkTTL (int networkTTL) [virtual]`
- 6.574.2.10** `virtual std::string activemq::commands::NetworkBridgeFilter::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 767).

6.574.3 Field Documentation

- 6.574.3.1** `const unsigned char activemq::commands::NetworkBridgeFilter::ID_NETWORKBRIDGEFILTER = 91 [static]`
- 6.574.3.2** `Pointer<BrokerId> activemq::commands::NetworkBridgeFilter::networkBrokerId [protected]`
- 6.574.3.3** `int activemq::commands::NetworkBridgeFilter::networkTTL [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/NetworkBridgeFilter.h`

6.575 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2617).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.575.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2617).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.575.2 Constructor & Destructor Documentation

6.575.2.1 `activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller () [inline]`

6.575.2.2 `virtual activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller () [inline, virtual]`

6.575.3 Member Function Documentation

6.575.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1505).

6.575.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1511).

6.575.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1518).

6.575.3.4 virtual void **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure** *
dataStructure, **decaf::io::DataInputStream** * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.575.3.5 virtual int **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **utils::BooleanStream** * *bs*) throw (
decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.575.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal2`
`(OpenWireFormat * wireFormat, commands::DataStructure`
`* dataStructure, decaf::io::DataOutputStream * dataOut,`
`utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.575.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightUnmarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure`
`* dataStructure, decaf::io::DataInputStream * dataIn,`
`utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h`

6.576 `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilter` Class Reference

Marshaling code for Open Wire Format for `NetworkBridgeFilterMarshaller` (p. 2620).

#include <src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.576.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2620).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.576.2 Constructor & Destructor Documentation

6.576.2.1 `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller () [inline]`

6.576.2.2 `virtual activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller () [inline, virtual]`

6.576.3 Member Function Documentation

6.576.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.576.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.576.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.576.3.4 virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.576.3.5 virtual int activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.576.3.6 virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.576.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h`

6.577 **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller** Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2624).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.577.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2624).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.577.2 Constructor & Destructor Documentation

6.577.2.1 `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller () [inline]`

6.577.2.2 `virtual activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller () [inline, virtual]`

6.577.3 Member Function Documentation

6.577.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.577.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.577.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.577.3.4 virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.577.3.5 virtual int activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.577.3.6 virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.577.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h`

6.578 **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller** Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2628).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.578.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2628).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.578.2 Constructor & Destructor Documentation

6.578.2.1 `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller () [inline]`

6.578.2.2 `virtual activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller () [inline, virtual]`

6.578.3 Member Function Documentation

6.578.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.578.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.578.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.578.3.4 virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.578.3.5 virtual int activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.578.3.6 virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.578.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h

6.579 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2632).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.579.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2632).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.579.2 Constructor & Destructor Documentation

6.579.2.1 `activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller () [inline]`

6.579.2.2 `virtual activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller () [inline, virtual]`

6.579.3 Member Function Documentation

6.579.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.579.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.579.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.579.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.579.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.579.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.579.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h`

6.580 **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller** Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2636).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.580.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2636).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.580.2 Constructor & Destructor Documentation

6.580.2.1 `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller () [inline]`

6.580.2.2 `virtual activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller () [inline, virtual]`

6.580.3 Member Function Documentation

6.580.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.580.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.580.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.580.3.4 virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.580.3.5 virtual int activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.580.3.6 virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.580.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h

6.581 decaf::net::NoRouteToHostException Class Reference

```
#include <src/main/decaf/net/NoRouteToHostException.h>
```

Inheritance diagram for decaf::net::NoRouteToHostException:

Public Member Functions

- **NoRouteToHostException** () throw ()
Default Constructor.

- **NoRouteToHostException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **NoRouteToHostException** (const NoRouteToHostException &ex) throw ()
Copy Constructor.
- **NoRouteToHostException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoRouteToHostException** (const std::exception *cause) throw ()
Constructor.
- **NoRouteToHostException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoRouteToHostException * clone** () const
Clones this exception.
- virtual **~NoRouteToHostException** () throw ()

6.581.1 Constructor & Destructor Documentation

6.581.1.1 decaf::net::NoRouteToHostException::NoRouteToHostException () throw () [inline]

Default Constructor.

6.581.1.2 decaf::net::NoRouteToHostException::NoRouteToHostException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.581.1.3 decaf::net::NoRouteToHostException::NoRouteToHostException (const NoRouteToHostException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.581.1.4 `decaf::net::NoRouteToHostException::NoRouteToHostException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.581.1.5 `decaf::net::NoRouteToHostException::NoRouteToHostException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.581.1.6 `decaf::net::NoRouteToHostException::NoRouteToHostException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.581.1.7 `virtual decaf::net::NoRouteToHostException::~~NoRouteToHostException () throw () [inline, virtual]`

6.581.2 Member Function Documentation

6.581.2.1 `virtual NoRouteToHostException* decaf::net::NoRouteToHostException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3300).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/NoRouteToHostException.h`

6.582 decaf::security::NoSuchAlgorithmException Class Reference

```
#include <src/main/decaf/security/NoSuchAlgorithmException.h>
```

Inheritance diagram for decaf::security::NoSuchAlgorithmException:

Public Member Functions

- **NoSuchAlgorithmException** () throw ()
Default Constructor.
- **NoSuchAlgorithmException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **NoSuchAlgorithmException** (const NoSuchAlgorithmException &ex) throw ()
Copy Constructor.
- **NoSuchAlgorithmException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchAlgorithmException** (const std::exception *cause) throw ()
Constructor.
- **NoSuchAlgorithmException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchAlgorithmException * clone** () const
Clones this exception.
- virtual **~NoSuchAlgorithmException** () throw ()

6.582.1 Constructor & Destructor Documentation

6.582.1.1 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException () throw () [inline]`

Default Constructor.

6.582.1.2 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.582.1.3 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const NoSuchAlgorithmException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.582.1.4 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.582.1.5 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.582.1.6 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException`
`(const char * file, const int lineNumber, const char * msg, ...)`
`throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.582.1.7 `virtual`
`decaf::security::NoSuchAlgorithmException::~~NoSuchAlgorithmException`
`() throw ()` [inline, virtual]

6.582.2 Member Function Documentation

6.582.2.1 `virtual NoSuchAlgorithmException* de-`
`caf::security::NoSuchAlgorithmException::clone ()`
`const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p.1847).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchAlgorithmException.h`

6.583 decaf::lang::exceptions::NoSuchElementException Class Reference

```
#include <src/main/decaf/lang/exceptions/NoSuchElementException.h>
```

Inheritance diagram for `decaf::lang::exceptions::NoSuchElementException`:

Public Member Functions

- **NoSuchElementException** () throw ()
Default Constructor.
- **NoSuchElementException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1712).*
- **NoSuchElementException** (const **NoSuchElementException** &ex) throw ()
Copy Constructor.
- **NoSuchElementException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchElementException** (const std::exception *cause) throw ()
Constructor.
- **NoSuchElementException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchElementException** * clone () const
Clones this exception.
- virtual ~**NoSuchElementException** () throw ()

6.583.1 Constructor & Destructor Documentation

6.583.1.1 **decaf::lang::exceptions::NoSuchElementException::NoSuchElementException**
() throw () [inline]

Default Constructor.

6.583.1.2 **decaf::lang::exceptions::NoSuchElementException::NoSuchElementException**
(const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.583.1.3 **decaf::lang::exceptions::NoSuchElementException::NoSuchElementException**
(const **NoSuchElementException** & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.583.1.4 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException
(const char * *file*, const int *lineNumber*, const std::exception *
cause, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.583.1.5 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException
(const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause **Pointer** (p.2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.583.1.6 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException
(const char * *file*, const int *lineNumber*, const char * *msg*, ...)
throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.583.1.7 virtual
decaf::lang::exceptions::NoSuchElementException::~~NoSuchElementException
() throw () [inline, virtual]

6.583.2 Member Function Documentation

6.583.2.1 virtual NoSuchElementException* de-
caf::lang::exceptions::NoSuchElementException::clone (
) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p.1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/NoSuchElementException.h`

6.584 decaf::security::NoSuchProviderException Class Reference

```
#include <src/main/decaf/security/NoSuchProviderException.h>
```

Inheritance diagram for `decaf::security::NoSuchProviderException`:

Public Member Functions

- **NoSuchProviderException** () throw ()
Default Constructor.
- **NoSuchProviderException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **NoSuchProviderException** (const **NoSuchProviderException** &ex) throw ()
Copy Constructor.
- **NoSuchProviderException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchProviderException** (const std::exception *cause) throw ()
Constructor.
- **NoSuchProviderException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchProviderException** * clone () const
Clones this exception.
- virtual ~**NoSuchProviderException** () throw ()

6.584.1 Constructor & Destructor Documentation

6.584.1.1 decaf::security::NoSuchProviderException::NoSuchProviderException () throw () [inline]

Default Constructor.

6.584.1.2 decaf::security::NoSuchProviderException::NoSuchProviderException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.584.1.3 decaf::security::NoSuchProviderException::NoSuchProviderException (const NoSuchProviderException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.584.1.4 decaf::security::NoSuchProviderException::NoSuchProviderException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.584.1.5 decaf::security::NoSuchProviderException::NoSuchProviderException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.584.1.6 `decaf::security::NoSuchProviderException::NoSuchProviderException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.584.1.7 `virtual decaf::security::NoSuchProviderException::~~NoSuchProviderException () throw ()` [inline, virtual]

6.584.2 Member Function Documentation

6.584.2.1 `virtual NoSuchProviderException* decaf::security::NoSuchProviderException::clone () const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p.1847).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchProviderException.h`

6.585 decaf::lang::exceptions::NullPointerException Class Reference

```
#include <src/main/decaf/lang/exceptions/NullPointerException.h>
```

Inheritance diagram for `decaf::lang::exceptions::NullPointerException`:

Public Member Functions

- **NullPointerException** () throw ()
Default Constructor.
- **NullPointerException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1712).*
- **NullPointerException** (const **NullPointerException** &ex) throw ()
Copy Constructor.
- **NullPointerException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NullPointerException** (const std::exception *cause) throw ()
Constructor.
- **NullPointerException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NullPointerException** * **clone** () const
Clones this exception.
- virtual ~**NullPointerException** () throw ()

6.585.1 Constructor & Destructor Documentation

6.585.1.1 decaf::lang::exceptions::NullPointerException::NullPointerException () throw () [inline]

Default Constructor.

6.585.1.2 decaf::lang::exceptions::NullPointerException::NullPointerException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.585.1.3 decaf::lang::exceptions::NullPointerException::NullPointerException (const NullPointerException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.585.1.4 `decaf::lang::exceptions::NullPointerException::NullPointerException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.585.1.5 `decaf::lang::exceptions::NullPointerException::NullPointerException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause **Pointer** (p.2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.585.1.6 `decaf::lang::exceptions::NullPointerException::NullPointerException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.585.1.7 `virtual decaf::lang::exceptions::NullPointerException::~~NullPointerException () throw () [inline, virtual]`

6.585.2 Member Function Documentation

6.585.2.1 `virtual NullPointerException* decaf::lang::exceptions::NullPointerException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/NullPointerException.h`

6.586 decaf::lang::Number Class Reference

The abstract class **Number** (p. 2653) is the superclass of classes **Byte** (p. 884), **Double** (p. 1672), **Float** (p. 1780), **Integer** (p. 1941), **Long** (p. 2267), and **Short** (p. 3220).

```
#include <src/main/decaf/lang/Number.h>
```

Inheritance diagram for decaf::lang::Number:

Public Member Functions

- virtual `~Number ()`
- virtual unsigned char **byteValue ()** const
Answers the byte value which the receiver represents.
- virtual double **doubleValue ()** const =0
Answers the double value which the receiver represents.
- virtual float **floatValue ()** const =0
Answers the float value which the receiver represents.
- virtual int **intValue ()** const =0
Answers the int value which the receiver represents.
- virtual long long **longValue ()** const =0
Answers the long value which the receiver represents.
- virtual short **shortValue ()** const
Answers the short value which the receiver represents.

6.586.1 Detailed Description

The abstract class **Number** (p. 2653) is the superclass of classes **Byte** (p. 884), **Double** (p. 1672), **Float** (p. 1780), **Integer** (p. 1941), **Long** (p. 2267), and **Short** (p. 3220). Subclasses of **Number** (p. 2653) must provide methods to convert the represented numeric value to byte, double, float, int, long, and short.

6.586.2 Constructor & Destructor Documentation

6.586.2.1 `virtual decaf::lang::Number::~~Number () [inline, virtual]`

6.586.3 Member Function Documentation

6.586.3.1 `virtual unsigned char decaf::lang::Number::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

6.586.3.2 `virtual double decaf::lang::Number::doubleValue () const [pure virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implemented in `decaf::util::concurrent::atomic::AtomicInteger` (p. 682).

6.586.3.3 `virtual float decaf::lang::Number::floatValue () const [pure virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implemented in `decaf::util::concurrent::atomic::AtomicInteger` (p. 682).

6.586.3.4 `virtual int decaf::lang::Number::intValue () const [pure virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implemented in `decaf::util::concurrent::atomic::AtomicInteger` (p. 684).

6.586.3.5 `virtual long long decaf::lang::Number::longValue () const [pure virtual]`

Answers the long value which the receiver represents.

Returns

long long the value of the receiver.

Implemented in `decaf::util::concurrent::atomic::AtomicInteger` (p. 684).

6.586.3.6 virtual short decaf::lang::Number::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/NumberFormatException`

6.587 decaf::lang::exceptions::NumberFormatException Class Reference

```
#include <src/main/decaf/lang/exceptions/NumberFormatException.h>
```

Inheritance diagram for `decaf::lang::exceptions::NumberFormatException`:

Public Member Functions

- **NumberFormatException** ()
Default Constructor.
- **NumberFormatException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1712).*
- **NumberFormatException** (const **NumberFormatException** &ex) throw ()
Copy Constructor.
- **NumberFormatException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NumberFormatException** (const std::exception *cause) throw ()
Constructor.
- **NumberFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.

- virtual **NumberFormatException** * **clone** () const
Clones this exception.
- virtual ~**NumberFormatException** () throw ()

6.587.1 Constructor & Destructor Documentation

6.587.1.1 `decaf::lang::exceptions::NumberFormatException::NumberFormatException () [inline]`

Default Constructor.

Referenced by clone().

6.587.1.2 `decaf::lang::exceptions::NumberFormatException::NumberFormatException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.587.1.3 `decaf::lang::exceptions::NumberFormatException::NumberFormatException (const NumberFormatException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.587.1.4 `decaf::lang::exceptions::NumberFormatException::NumberFormatException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

References decaf::lang::Exception::buildMessage(), and decaf::lang::Exception::setMark().

6.587.1.5 decaf::lang::exceptions::NumberFormatException::NumberFormatException
(const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

***cause* Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.587.1.6 decaf::lang::exceptions::NumberFormatException::NumberFormatException
(const char * *file*, const int *lineNumber*, const char * *msg*, ...)
throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

References decaf::lang::Exception::buildMessage(), and decaf::lang::Exception::setMark().

6.587.1.7 virtual
decaf::lang::exceptions::NumberFormatException::~~NumberFormatException
() throw () [inline, virtual]

6.587.2 Member Function Documentation

6.587.2.1 virtual NumberFormatException* decaf::lang::exceptions::NumberFormatException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

References NumberFormatException().

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/NumberFormatException.h

6.588 cms::ObjectMessage Class Reference

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

```
#include <src/main/cms/ObjectMessage.h>
```

Inheritance diagram for cms::ObjectMessage:

Public Member Functions

- virtual `~ObjectMessage()`

6.588.1 Detailed Description

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object. serialized `ObjectMessage` (p. 2658)s.

Since

1.0

6.588.2 Constructor & Destructor Documentation

6.588.2.1 virtual `cms::ObjectMessage::~ObjectMessage()` [`inline`, `virtual`]

The documentation for this class was generated from the following file:

- `src/main/cms/ObjectMessage.h`

6.589 decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference

Provides an SSLContext that wraps the OpenSSL API.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLContextSpi:

Public Member Functions

- `OpenSSLContextSpi()`
- virtual `~OpenSSLContextSpi()`
- virtual void `providerInit(security::SecureRandom *random)`

Perform the initialization of this Context.

Parameters

random Pointer to an instance of a secure random number generator.

Exceptions

NullPointerException if the *SecureRandom* instance is *NULL*.
KeyManagementException if an error occurs while initializing the context.

- virtual **decaf::net::SocketFactory * providerGetSocketFactory ()**

Returns a **SocketFactory** (p. 3301) instance that can be used to create new **SSLSocket** (p. 3337) objects.

The **SocketFactory** (p. 3301) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 3301) instance that can be used to create new *SSLSockets*.

Exceptions

IllegalStateException if the **SSLContextSpi** (p. 3324) object requires initialization but has not been initialized yet.

- virtual **decaf::net::ServerSocketFactory * providerGetServerSocketFactory ()**

Returns a **ServerSocketFactory** (p. 3145) instance that can be used to create new **SSLServerSocket** (p. 3329) objects.

The **ServerSocketFactory** (p. 3145) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 3301) instance that can be used to create new *SSLServerSockets*.

Exceptions

IllegalStateException if the **SSLContextSpi** (p. 3324) object requires initialization but has not been initialized yet.

Friends

- class **OpenSSLSocket**
- class **OpenSSLSocketFactory**

6.589.1 Detailed Description

Provides an *SSLContext* that wraps the *OpenSSL* API.

Since

1.0

6.589.2 Constructor & Destructor Documentation

6.589.2.1 `decaf::internal::net::ssl::openssl::OpenSSLContextSpi::OpenSSLContextSpi
()`

6.589.2.2 `virtual
decaf::internal::net::ssl::openssl::OpenSSLContextSpi::~~OpenSSLContextSpi
() [virtual]`

6.589.3 Member Function Documentation

6.589.3.1 `virtual decaf::net::ServerSocketFactory* de-
caf::internal::net::ssl::openssl::OpenSSLContextSpi::providerGetServerSocketFactory
() [virtual]`

Returns a **ServerSocketFactory** (p. 3145) instance that can be used to create new **SSLServerSocket** (p. 3329) objects.

The **ServerSocketFactory** (p. 3145) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 3301) instance that can be used to create new SSLServerSockets.

Exceptions

IllegalStateException if the **SSLContextSpi** (p. 3324) object requires initialization but has not been initialized yet.

Implements **decaf::net::ssl::SSLContextSpi** (p. 3325).

6.589.3.2 `virtual decaf::net::SocketFactory* de-
caf::internal::net::ssl::openssl::OpenSSLContextSpi::providerGetSocketFactory
() [virtual]`

Returns a **SocketFactory** (p. 3301) instance that can be used to create new **SSLSocket** (p. 3337) objects.

The **SocketFactory** (p. 3301) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 3301) instance that can be used to create new SSLSockets.

Exceptions

IllegalStateException if the **SSLContextSpi** (p. 3324) object requires initialization but has not been initialized yet.

Implements **decaf::net::ssl::SSLContextSpi** (p. 3325).

6.589.3.3 virtual void decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerInit (security::SecureRandom * *random*) [virtual]

Perform the initialization of this Context.

Parameters

random Pointer to an instance of a secure random number generator.

Exceptions

NullPointerException if the SecureRandom instance is NULL.

KeyManagementException if an error occurs while initializing the context.

Implements decaf::net::ssl::SSLContextSpi (p. 3326).

6.589.4 Friends And Related Function Documentation

6.589.4.1 friend class OpenSSLSocket [friend]

6.589.4.2 friend class OpenSSLSocketFactory [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h

6.590 decaf::internal::net::ssl::openssl::OpenSSLParameters Class Reference

Container class for parameters that are Common to OpenSSL socket classes.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h>
```

Public Member Functions

- virtual ~OpenSSLParameters ()
- bool getNeedClientAuth () const
- void setNeedClientAuth (bool value)
- bool getWantClientAuth () const
- void setWantClientAuth (bool value)
- bool getUseClientMode () const
- void setUseClientMode (bool value)
- std::vector< std::string > getSupportedCipherSuites () const
- std::vector< std::string > getSupportedProtocols () const
- std::vector< std::string > getEnabledCipherSuites () const
- void setEnabledCipherSuites (const std::vector< std::string > &suites)
- std::vector< std::string > getEnabledProtocols () const
- void setEnabledProtocols (const std::vector< std::string > &protocols)
- OpenSSLParameters * clone () const

Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL_CTX as this object's.

6.590.1 Detailed Description

Container class for parameters that are Common to OpenSSL socket classes.

Since

1.0

6.590.2 Constructor & Destructor Documentation

6.590.2.1 `virtual
decaf::internal::net::ssl::openssl::OpenSSLParameters::~~OpenSSLParameters
() [virtual]`

6.590.3 Member Function Documentation

6.590.3.1 `OpenSSLParameters* de-
caf::internal::net::ssl::openssl::OpenSSLParameters::clone () const`

Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL_CTX as this object's.

- 6.590.3.2 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledCipherSuites () const`
- 6.590.3.3 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledProtocols () const`
- 6.590.3.4 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getNeedClientAuth () const [inline]`
- 6.590.3.5 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedCipherSuites () const`
- 6.590.3.6 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedProtocols () const`
- 6.590.3.7 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getUseClientMode () const [inline]`
- 6.590.3.8 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getWantClientAuth () const [inline]`
- 6.590.3.9 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledCipherSuites (const std::vector< std::string > & suites)`
- 6.590.3.10 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledProtocols (const std::vector< std::string > & protocols)`
- 6.590.3.11 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setNeedClientAuth (bool value) [inline]`
- 6.590.3.12 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setUseClientMode (bool value) [inline]`
- 6.590.3.13 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setWantClientAuth (bool value) [inline]`

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h`

6.591 decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference

SSLServerSocket based on OpenSSL library code.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLServerSocket:

Public Member Functions

- **OpenSSLServerSocket** (**OpenSSLParameters** *parameters)
- virtual **~OpenSSLServerSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () const
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3329).*
*Normally not all of these cipher suites will be enabled on the **Socket** (p. 3281).*
Returns
a vector containing the names of all the supported cipher suites.
- virtual std::vector< std::string > **getSupportedProtocols** () const
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3329) instance.*
Returns
a vector containing the names of all the supported protocols.
- virtual std::vector< std::string > **getEnabledCipherSuites** () const
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3329).*
Returns
vector of the names of all enabled Cipher Suites.
- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)
*Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3329) connection. Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.*
Parameters
suites An Vector of names for all the Cipher Suites that are to be enabled.
Exceptions
***IllegalArgumentException** if the vector is empty or one of the names is invalid.*
- virtual std::vector< std::string > **getEnabledProtocols** () const
*Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3329).*
Returns
vector of the names of all enabled Protocols.

- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)
*Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3329) connection. Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.*
Parameters
protocols An Vector of names for all the Protocols that are to be enabled.
Exceptions
IllegalArgumentException if the vector is empty or one of the names is invalid.
- virtual bool **getWantClientAuth** () const
Returns
*true if the **Socket** (p. 3281) request client Authentication.*
- virtual void **setWantClientAuth** (bool value)
*Sets whether or not this **Socket** (p. 3281) will request Client Authentication. If set to true the **Socket** (p. 3281) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.*
Parameters
value Whether the server socket should request client authentication.
- virtual bool **getNeedClientAuth** () const
Returns
*true if the **Socket** (p. 3281) requires client Authentication.*
- virtual void **setNeedClientAuth** (bool value)
*Sets whether or not this **Socket** (p. 3281) will require Client Authentication. If set to true the **Socket** (p. 3281) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.*
Parameters
value Whether the server socket should require client authentication.
- virtual **decaf::net::Socket * accept** () throw (decaf::io::IOException)
*Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3136), the caller blocks until a connection is made. If the *SO_TIMEOUT* option is set this method could throw a **SocketTimeoutException** (p. 3319) if the operation times out.*
Returns
*a new **Socket** (p. 3281) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.*
Exceptions
IOException if an I/O error occurs while binding the socket.
SocketException (p. 3298) if an error occurs while blocking on the accept call.
SocketTimeoutException (p. 3319) if the *SO_TIMEOUT* option was used and the accept timed out.

6.591.1 Detailed Description

SSLServerSocket based on OpenSSL library code.

Since

1.0

6.591.2 Constructor & Destructor Documentation

6.591.2.1 `decaf::internal::net::ssl::openssl::OpenSSLServerSocket::OpenSSLServerSocket (OpenSSLParameters * parameters)`

6.591.2.2 `virtual decaf::internal::net::ssl::openssl::OpenSSLServerSocket::~~OpenSSLServerSocket () [virtual]`

6.591.3 Member Function Documentation

6.591.3.1 `virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLServerSocket::accept () throw (decaf::io::IOException) [virtual]`

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3136), the caller blocks until a connection is made.

If the SO_TIMEOUT option is set this method could throw a **SocketTimeoutException** (p. 3319) if the operation times out.

Returns

a new **Socket** (p. 3281) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

Exceptions

IOException if an I/O error occurs while binding the socket.

SocketException (p. 3298) if an error occurs while blocking on the accept call.

SocketTimeoutException (p. 3319) if the SO_TIMEOUT option was used and the accept timed out.

Reimplemented from **decaf::net::ServerSocket** (p. 3141).

6.591.3.2 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledCipherSuites () const [virtual]`

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3329).

Returns

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3332).

6.591.3.3 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledProtocols () const [virtual]`

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3329).

Returns

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3332).

6.591.3.4 `virtual bool decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getNeedClientAuth () const [virtual]`

Returns

true if the **Socket** (p. 3281) requires client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3333).

6.591.3.5 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedCipherSuites () const [virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3329).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3281).

Returns

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3333).

6.591.3.6 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedProtocols () const [virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3329) instance.

Returns

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3333).

6.591.3.7 virtual bool decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getWantClientAuth () const [virtual]

Returns

true if the **Socket** (p. 3281) request client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3333).

6.591.3.8 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledCipherSuites (const std::vector< std::string > & *suites*) [virtual]

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3329) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

suites An Vector of names for all the Cipher Suites that are to be enabled.

Exceptions

IllegalArgumentException if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3333).

6.591.3.9 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledProtocols (const std::vector< std::string > & *protocols*) [virtual]

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3329) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

protocols An Vector of names for all the Protocols that are to be enabled.

Exceptions

IllegalArgumentException if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3334).

6.591.3.10 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setNeedClientAuth (bool *value*) [virtual]

Sets whether or not this **Socket** (p. 3281) will require Client Authentication.

If set to true the **Socket** (p. 3281) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

Parameters

value Whether the server socket should require client authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3334).

6.591.3.11 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setWantClientAuth (bool *value*) [virtual]

Sets whether or not this **Socket** (p. 3281) will request Client Authentication.

If set to true the **Socket** (p. 3281) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

Parameters

value Whether the server socket should request client authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3334).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLServerSocket.h**

6.592 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory:

Public Member Functions

- **OpenSSLServerSocketFactory (OpenSSLContextSpi *parent)**
- **virtual ~OpenSSLServerSocketFactory ()**
- **virtual decaf::net::ServerSocket * createServerSocket ()**

*Create a new **ServerSocket** (p. 3136) that is unbound.*

*The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.*

Returns

*new **ServerSocket** (p. 3136) pointer that is owned by the caller.*

Exceptions

***IOException** if the **ServerSocket** (p. 3136) cannot be created for some reason.*

- **virtual decaf::net::ServerSocket * createServerSocket (int port)**

*Create a new **ServerSocket** (p. 3136) that is bound to the given port.*

*The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.*

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

- virtual **decaf::net::ServerSocket * createServerSocket** (int port, int backlog)

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will use the specified connection backlog setting.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

- virtual **decaf::net::ServerSocket * createServerSocket** (int port, int backlog, const **decaf::net::InetAddress *address**)

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is **NULL** than the **ServerSocket** (p. 3136) will listen on all interfaces.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

address The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

- virtual **std::vector< std::string > getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 3336)

- virtual **std::vector< std::string > getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 3336)

6.592.1 Detailed Description

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

Since

1.0

6.592.2 Constructor & Destructor Documentation

6.592.2.1 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::OpenSSLServerSocketFactory (OpenSSLContextSpi * *parent*)

6.592.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::~OpenSSLServerSocketFactory () [virtual]

6.592.3 Member Function Documentation

6.592.3.1 virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket () [virtual]

Create a new **ServerSocket** (p. 3136) that is unbound.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3146).

6.592.3.2 virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket (int *port*) [virtual]

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3147).

6.592.3.3 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket (int port, int backlog, const decaf::net::InetAddress * address) [virtual]`

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3136) will listen on all interfaces.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

address The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3147).

6.592.3.4 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket (int port, int backlog) [virtual]`

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will use the specified connection backlog setting.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3147).

6.592.3.5 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getDefaultCipherSuites () [virtual]`

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 3336)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 3336).

6.592.3.6 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getSupportedCipherSuites () [virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 3336)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 3336).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h`

6.593 decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference

Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocket:

Public Member Functions

- **OpenSSLSocket** (**OpenSSLParameters** *parameters)
- **OpenSSLSocket** (**OpenSSLParameters** *parameters, const **decaf::net::InetAddress** *address, int port)
- **OpenSSLSocket** (**OpenSSLParameters** *parameters, const **decaf::net::InetAddress** *address, int port, const **decaf::net::InetAddress** *localAddress, int localPort)
- **OpenSSLSocket** (**OpenSSLParameters** *parameters, const std::string &host, int port)
- **OpenSSLSocket** (**OpenSSLParameters** *parameters, const std::string &host, int port, const **decaf::net::InetAddress** *localAddress, int localPort)
- virtual ~**OpenSSLSocket** ()
- virtual void **connect** (const std::string &host, int port, int timeout) throw (**decaf::io::IOException**, **decaf::lang::exceptions::IllegalArgumentException**)

Connects to the specified destination, with a specified timeout value.

*If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3319) is thrown. A timeout value of zero is treated as an infinite timeout.*

Parameters

host The host name or IP address of the remote host to connect to.

port The port on the remote host to connect to.

timeout The number of Milliseconds to wait before treating the connection as failed.

Exceptions

IOException Thrown if a failure occurred in the connect.

SocketTimeoutException (p. 3319) if the timeout for connection is exceeded.

IllegalArgumentException if the timeout value is negative or the endpoint is invalid.

- virtual void **close** () throw (**decaf::io::IOException**)

*Closes the **Socket** (p. 3281).*

*Once closed a **Socket** (p. 3281) cannot be connected or otherwise operated upon, a new **Socket** (p. 3281) instance must be created.*

Exceptions

IOException if an I/O error occurs while closing the **Socket** (p. 3281).

- virtual **decaf::io::InputStream** * **getInputStream** () throw (**decaf::io::IOException**)

*Gets the **InputStream** for this socket if its connected.*

*The pointer returned is the property of the associated **Socket** (p. 3281) and should not be deleted by the caller.*

*When the returned **InputStream** is performing a blocking operation and the underlying connection is closed or otherwise broker the read calls will normally throw an exception to indicate the failure. Closing the **InputStream** will also close the underlying **Socket** (p. 3281).*

Returns

*The **InputStream** for this socket.*

Exceptions

IOException if an error occurs during creation of the **InputStream**, also if the **Socket** (p. 3281) is not connected or the input has been shutdown previously.

- virtual **decaf::io::OutputStream * getOutputStream ()** throw (decaf::io::IOException)

Gets the OutputStream for this socket if it is connected.

*The pointer returned is the property of the **Socket** (p. 3281) instance and should not be deleted by the caller.*

*Closing the returned **Socket** (p. 3281) will also close the underlying **Socket** (p. 3281).*

Returns

the OutputStream for this socket.

Exceptions

IOException *if an error occurs during the creation of this OutputStream, or if the **Socket** (p. 3281) is closed or the output has been shutdown previously.*

- virtual void **shutdownInput ()** throw (decaf::io::IOException)

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

Exceptions

IOException *if an I/O error occurs while performing this operation.*

- virtual void **shutdownOutput ()** throw (decaf::io::IOException)

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OutputStream::write will throw an IOException.

Exceptions

IOException *if an I/O error occurs while performing this operation.*

- virtual void **setOOBInline** (bool value) throw (decaf::net::SocketException)

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the Socket's InputStream, no notification is give.

Returns

true if OOBINLINE is enabled, false otherwise.

Exceptions

SocketException (p. 3298) *if an error is encountered while performing this operation.*

- virtual void **sendUrgentData** (int data) throw (decaf::io::IOException)

*Sends on byte of urgent data to the **Socket** (p. 3281).*

Parameters

data *The value to write as urgent data, only the lower eight bits are sent.*

Exceptions

IOException *if an I/O error occurs while performing this operation.*

- virtual std::vector< std::string > **getSupportedCipherSuites ()** const

*Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3337).*

*Normally not all of these cipher suites will be enabled on the **Socket** (p. 3281).*

Returns

a vector containing the names of all the supported cipher suites.

- virtual std::vector< std::string > **getSupportedProtocols ()** const

*Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3337) instance.*

Returns

a vector containing the names of all the supported protocols.

- virtual std::vector< std::string > **getEnabledCipherSuites** () const

*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3281).*

Returns

vector of the names of all enabled Cipher Suites.

- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)

*Sets the Cipher Suites that are to be enabled on the **SSL Socket** (p. 3281) connection. Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.*

Parameters

***suites** An Vector of names for all the Cipher Suites that are to be enabled.*

Exceptions

***IllegalArgumentException** if the vector is empty or one of the names is invalid.*

- virtual std::vector< std::string > **getEnabledProtocols** () const

*Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3281).*

Returns

vector of the names of all enabled Protocols.

- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)

*Sets the Protocols that are to be enabled on the **SSL Socket** (p. 3281) connection. Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.*

Parameters

***protocols** An Vector of names for all the Protocols that are to be enabled.*

Exceptions

***IllegalArgumentException** if the vector is empty or one of the names is invalid.*

- virtual void **startHandshake** ()

*Initiates a handshake for this **SSL Connection**, this can be necessary for several reasons such as using new encryption keys, or starting a new session.*

*When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not require to support multiple handshakes and can throw an **IOException** to indicate an error.*

Exceptions

***IOException** if an I/O error occurs while performing the Handshake*

- virtual void **setUseClientMode** (bool value)

Determines the mode that the socket uses when a handshake is initiated, client or server.

*This method must be called prior to any handshake attempts on this **Socket** (p. 3281), once a handshake has been initiated this socket remains the the set mode; client or server, for the life of this object.*

Parameters

value The mode setting, true for client or false for server.

Exceptions

IllegalArgumentException if the handshake process has begun and mode is locked.

- virtual bool **getUseClientMode** () const

*Gets whether this **Socket** (p. 3281) is in Client or Server mode, true indicates that the mode is set to Client.*

Returns

*true if the **Socket** (p. 3281) is in Client mode, false otherwise.*

- virtual void **setNeedClientAuth** (bool value)

*Sets the **Socket** (p. 3281) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

Parameters

value The value indicating if a client is required to authenticate itself or not.

- virtual bool **getNeedClientAuth** () const

Returns if this socket is configured to require client authentication, true means that is has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

- virtual void **setWantClientAuth** (bool value)

*Sets the **Socket** (p. 3281) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the setNeedClientAuth method.

Parameters

value The value indicating if a client is requested to authenticate itself or not.

- virtual bool **getWantClientAuth** () const

Returns if this socket is configured to request client authentication, true means that is has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

- int **read** (unsigned char *buffer, int size, int offset, int length)

Reads the requested data from the Socket and write it into the passed in buffer.

- void **write** (const unsigned char *buffer, int size, int offset, int length)
Writes the specified data in the passed in buffer to the Socket.
- int **available** ()
Gets the number of bytes in the Socket buffer that can be read without blocking.

6.593.1 Detailed Description

Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

Since

1.0

6.593.2 Constructor & Destructor Documentation

- 6.593.2.1** `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters * parameters)`
- 6.593.2.2** `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters * parameters, const decaf::net::InetAddress * address, int port)`
- 6.593.2.3** `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters * parameters, const decaf::net::InetAddress * address, int port, const decaf::net::InetAddress * localAddress, int localPort)`
- 6.593.2.4** `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters * parameters, const std::string & host, int port)`
- 6.593.2.5** `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters * parameters, const std::string & host, int port, const decaf::net::InetAddress * localAddress, int localPort)`
- 6.593.2.6** `virtual decaf::internal::net::ssl::openssl::OpenSSLSocket::~~OpenSSLSocket () [virtual]`

6.593.3 Member Function Documentation

- 6.593.3.1** `int decaf::internal::net::ssl::openssl::OpenSSLSocket::available ()`

Gets the number of bytes in the Socket buffer that can be read without blocking.

Returns

the number of bytes that can be read from the Socket without blocking.

Exceptions

IOException if an I/O error occurs while performing this operation.

6.593.3.2 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::close ()
throw (decaf::io::IOException) [virtual]`

Closes the **Socket** (p. 3281).

Once closed a **Socket** (p. 3281) cannot be connected or otherwise operated upon, a new **Socket** (p. 3281) instance must be created.

Exceptions

IOException if an I/O error occurs while closing the **Socket** (p. 3281).

Reimplemented from **decaf::net::Socket** (p. 3287).

6.593.3.3 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::connect
(const std::string & host, int port, int timeout) throw (
decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException
) [virtual]`

Connects to the specified destination, with a specified timeout value.

If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3319) is thrown. A timeout value of zero is treated as an infinite timeout.

Parameters

host The host name or IP address of the remote host to connect to.

port The port on the remote host to connect to.

timeout The number of Milliseconds to wait before treating the connection as failed.

Exceptions

IOException Thrown if a failure occurred in the connect.

SocketTimeoutException (p. 3319) if the timeout for connection is exceeded.

IllegalArgumentException if the timeout value is negative or the endpoint is invalid.

Reimplemented from **decaf::net::Socket** (p. 3288).

6.593.3.4 `virtual std::vector<std::string> de-
caf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledCipherSuites (
) const [virtual]`

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3281).

Returns

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLSocket** (p. 3340).

6.593.3.5 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledProtocols () const [virtual]`

Returns a vector containing the names of all the currently enabled Protocols for this SSL **Socket** (p. 3281).

Returns

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 3341).

6.593.3.6 `virtual decaf::io::InputStream* decaf::internal::net::ssl::openssl::OpenSSLSocket::getInputStream () throw (decaf::io::IOException) [virtual]`

Gets the InputStream for this socket if its connected.

The pointer returned is the property of the associated **Socket** (p. 3281) and should not be deleted by the caller.

When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broken the read calls will normally throw an exception to indicate the failure.

Closing the InputStream will also close the underlying **Socket** (p. 3281).

Returns

The InputStream for this socket.

Exceptions

IOException if an error occurs during creation of the InputStream, also if the **Socket** (p. 3281) is not connected or the input has been shutdown previously.

Reimplemented from **decaf::net::Socket** (p. 3289).

6.593.3.7 `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getNeedClientAuth () const [virtual]`

Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 3341).

6.593.3.8 `virtual decaf::io::OutputStream* decaf::internal::net::ssl::openssl::OpenSSLSocket::getOutputStream () throw (decaf::io::IOException) [virtual]`

Gets the OutputStream for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 3281) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 3281) will also close the underlying **Socket** (p. 3281).

Returns

the OutputStream for this socket.

Exceptions

IOException if an error occurs during the creation of this OutputStream, or if the **Socket** (p. 3281) is closed or the output has been shutdown previously.

Reimplemented from **decaf::net::Socket** (p. 3290).

6.593.3.9 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedCipherSuites () const [virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3337).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3281).

Returns

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLSocket** (p. 3341).

6.593.3.10 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedProtocols () const [virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3337) instance.

Returns

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 3342).

6.593.3.11 `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getUseClientMode () const [virtual]`

Gets whether this **Socket** (p. 3281) is in Client or Server mode, true indicates that the mode is set to Client.

Returns

true if the **Socket** (p. 3281) is in Client mode, false otherwise.

Implements **decaf::net::ssl::SSLSocket** (p. 3342).

6.593.3.12 `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getWantClientAuth () const [virtual]`

Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 3342).

6.593.3.13 `int decaf::internal::net::ssl::openssl::OpenSSLSocket::read (unsigned char * buffer, int size, int offset, int length)`

Reads the requested data from the **Socket** and write it into the passed in buffer.

Parameters

buffer The buffer to read into

size The size of the specified buffer

offset The offset into the buffer where reading should start filling.

length The number of bytes past offset to fill with data.

Returns

the actual number of bytes read or -1 if at EOF.

Exceptions

IOException if an I/O error occurs during the read.

NullPointerException if buffer is Null.

IndexOutOfBoundsException if offset + length is greater than buffer size.

6.593.3.14 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::sendUrgentData (int data) throw (decaf::io::IOException) [virtual]`

Sends one byte of urgent data to the **Socket** (p. 3281).

Parameters

data The value to write as urgent data, only the lower eight bits are sent.

Exceptions

IOException if an I/O error occurs while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 3293).

6.593.3.15 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledCipherSuites (const std::vector< std::string > & suites) [virtual]`

Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 3281) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

suites An Vector of names for all the Cipher Suites that are to be enabled.

Exceptions

IllegalArgumentException if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLSocket** (p. 3342).

6.593.3.16 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledProtocols (const std::vector< std::string > & protocols) [virtual]`

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 3281) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

protocols An Vector of names for all the Protocols that are to be enabled.

Exceptions

IllegalArgumentException if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLSocket** (p. 3343).

6.593.3.17 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setNeedClientAuth (bool value) [virtual]`

Sets the **Socket** (p. 3281) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the `setWantClientAuth` method.

Parameters

value The value indicating if a client is required to authenticate itself or not.

Implements **decaf::net::ssl::SSLSocket** (p. 3343).

6.593.3.18 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setOOBInline (bool value) throw (decaf::net::SocketException) [virtual]`

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the Socket's InputStream, no notification is give.

Returns

true if OOBINLINE is enabled, false otherwise.

Exceptions

SocketException (p. 3298) if an error is encountered while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 3294).

6.593.3.19 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setUseClientMode (bool value) [virtual]`

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 3281), once a handshake has be initiated this socket remains the the set mode; client or server, for the life of this object.

Parameters

value The mode setting, true for client or false for server.

Exceptions

IllegalArgumentException if the handshake process has begun and mode is locked.

Implements **decaf::net::ssl::SSLSocket** (p. 3344).

6.593.3.20 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setWantClientAuth (bool value) [virtual]`

Sets the **Socket** (p. 3281) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the setNeedClientAuth method.

Parameters

value The value indicating if a client is requested to authenticate itself or not.

Implements **decaf::net::ssl::SSLSocket** (p. 3344).

6.593.3.21 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownInput ()
throw (decaf::io::IOException) [virtual]`

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

Exceptions

IOException if an I/O error occurs while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 3296).

6.593.3.22 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownOutput ()
throw (decaf::io::IOException) [virtual]`

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OutputStream::write will throw an IOException.

Exceptions

IOException if an I/O error occurs while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 3297).

6.593.3.23 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::startHandshake ()
[virtual]`

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not required to support multiple handshakes and can throw an IOException to indicate an error.

Exceptions

IOException if an I/O error occurs while performing the Handshake

Implements **decaf::net::ssl::SSLSocket** (p. 3344).

6.593.3.24 `void decaf::internal::net::ssl::openssl::OpenSSLSocket::write (const
unsigned char * buffer, int size, int offset, int length)`

Writes the specified data in the passed in buffer to the Socket.

Parameters

- buffer* The buffer to write to the socket.
- size* The size of the specified buffer.
- offset* The offset into the buffer where the data to write starts at.
- length* The number of bytes past offset to write.

Exceptions

- IOException* if an I/O error occurs during the write.
- NullPointerException* if buffer is Null.
- IndexOutOfBoundsException* if offset + length is greater than buffer size.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h`

6.594 decaf::internal::net::ssl::openssl::OpenSSLSocketException Class Reference

Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketException:

Public Member Functions

- **OpenSSLSocketException** () throw ()
*Creates an new **OpenSSLSocketException** (p. 2686) with default values.*
- **OpenSSLSocketException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **OpenSSLSocketException** (const **OpenSSLSocketException** &ex) throw ()
Copy Constructor.
- **OpenSSLSocketException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
*Create a new **OpenSSLSocketException** (p. 2686) and initializes the file name and line number where this message occurred.*
- **OpenSSLSocketException** (const std::exception *cause) throw ()
*Creates a new **OpenSSLSocketException** (p. 2686) with the passed exception set as the cause of this exception.*
- **OpenSSLSocketException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Create a new *OpenSSLSocketException* (p. 2686) and initializes the file name and line number where this message occurred.

- **OpenSSLSocketException** (const char *file, const int lineNumber) throw ()
*Create a new **OpenSSLSocketException** (p. 2686) and initializes the file name and line number where this message occurred.*
- virtual **OpenSSLSocketException** * clone () const
Clones this exception.
- virtual ~**OpenSSLSocketException** () throw ()

Protected Member Functions

- std::string **getErrorString** () const
Gets and formats an error message string from the OpenSSL error stack.

6.594.1 Detailed Description

Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.

Since

1.0

6.594.2 Constructor & Destructor Documentation

6.594.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException () throw ()

Creates an new **OpenSSLSocketException** (p. 2686) with default values.

6.594.2.2 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const Exception & ex) throw ()

Conversion Constructor from some other Exception.

Parameters

ex An Exception object that should become this type of Exception.

6.594.2.3 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const OpenSSLSocketException & ex) throw ()

Copy Constructor.

Parameters

ex The **OpenSSLSocketException** (p. 2686) whose values should be copied to this instance.

6.594.2.4 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()`

Create a new **OpenSSLSocketException** (p. 2686) and initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown (can be null).

msg The error message to report.

... The list of primitives that are formatted into the message.

6.594.2.5 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const std::exception * cause) throw ()`

Creates a new **OpenSSLSocketException** (p. 2686) with the passed exception set as the cause of this exception.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.594.2.6 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const char * file, const int lineNumber, const char * msg, ...) throw ()`

Create a new **OpenSSLSocketException** (p. 2686) and initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

Parameters

file The file name where exception occurs.

lineNumber The line number where the exception occurred.

msg The error message to report.

... The list of primitives that are formatted into the message

6.594.2.7 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const char * file, const int lineNumber) throw ()`

Create a new **OpenSSLSocketException** (p. 2686) and initializes the file name and line number where this message occurred.

Sets the message to report by getting the complete set of error messages from the OpenSSL error stack and concatenating them into one string.

Parameters

- file* The file name where exception occurs.
- lineNumber* The line number where the exception occurred.

6.594.2.8 **virtual**
decaf::internal::net::ssl::openssl::OpenSSLSocketException::~~OpenSSLSocketException
() throw () [virtual]

6.594.3 Member Function Documentation

6.594.3.1 **virtual OpenSSLSocketException* de-**
caf::internal::net::ssl::openssl::OpenSSLSocketException::clone () const
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override this method.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3300).

6.594.3.2 **std::string de-**
caf::internal::net::ssl::openssl::OpenSSLSocketException::getErrorString (
) const [protected]

Gets and formats an error message string from the OpenSSL error stack.

Returns

a string containing the complete OpenSSL error string.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h`

6.595 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference

Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketFactory:

Public Member Functions

- **OpenSSLSocketFactory** (**OpenSSLContextSpi** *parent)
- virtual **~OpenSSLSocketFactory** ()
- virtual **decaf::net::Socket** * **createSocket** () throw (decaf::io::IOException)

*Creates an unconnected **Socket** (p. 3281) object.*

Returns

*a new **Socket** (p. 3281) object, caller must free this object when done.*

Exceptions

***IOException** if the **Socket** (p. 3281) cannot be created.*
- virtual **decaf::net::Socket** * **createSocket** (const **decaf::net::InetAddress** *host, int port) throw (decaf::io::IOException, decaf::net::UnknownHostException)

*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*

Parameters

host The host to connect the socket to.
port The port on the remote host to connect to.

Returns

*a new **Socket** (p. 3281) object, caller must free this object when done.*

Exceptions

***IOException** if an I/O error occurs while creating the **Socket** (p. 3281) object.
UnknownHostException (p. 3649) if the host name is not known.*
- virtual **decaf::net::Socket** * **createSocket** (const **decaf::net::InetAddress** *host, int port, const **decaf::net::InetAddress** *ifAddress, int localPort) throw (decaf::io::IOException, decaf::net::UnknownHostException)

*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*

*The **Socket** (p. 3281) will be bound to the specified local address and port.*

Parameters

host The host to connect the socket to.
port The port on the remote host to connect to.
ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.
localPort The local port to bind the **Socket** (p. 3281) to.

Returns

*a new **Socket** (p. 3281) object, caller must free this object when done.*

Exceptions

***IOException** if an I/O error occurs while creating the **Socket** (p. 3281) object.
UnknownHostException (p. 3649) if the host name is not known.*
- virtual **decaf::net::Socket** * **createSocket** (const std::string &hostname, int port) throw (decaf::io::IOException, decaf::net::UnknownHostException)

*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*

Parameters

host The host name or IP address to connect the socket to.
port The port on the remote host to connect to.

Returns

*a new **Socket** (p. 3281) object, caller must free this object when done.*

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.
UnknownHostException (p. 3649) if the host name is not known.

- virtual **decaf::net::Socket * createSocket** (const std::string &name, int port, const **decaf::net::InetAddress *ifAddress**, int localPort) throw (**decaf::io::IOException**, **decaf::net::UnknownHostException**)

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host name or IP address to connect the socket to.
port The port on the remote host to connect to.
ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.
localPort The local port to bind the **Socket** (p. 3281) to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.
UnknownHostException (p. 3649) if the host name is not known.

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 3347)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 3347)

- virtual **decaf::net::Socket * createSocket** (**decaf::net::Socket *socket**, std::string host, int port, bool autoClose)

Returns a socket layered over an existing socket connected to the named host, at the given port. This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

socket The existing socket to layer over.
host The server host the original **Socket** (p. 3281) is connected to.
port The server port the original **Socket** (p. 3281) is connected to.
autoClose Should the layered over **Socket** (p. 3281) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 3281) instance that wraps the given **Socket** (p. 3281).

Exceptions

IOException if an I/O exception occurs while performing this operation.
UnknownHostException (p. 3649) if the host is unknown.

6.595.1 Detailed Description

Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

Since

1.0

6.595.2 Constructor & Destructor Documentation

6.595.2.1 `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::OpenSSLSocketFactory (OpenSSLContextSpi * parent)`

6.595.2.2 `virtual decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::~~OpenSSLSocketFactory () [virtual]`

6.595.3 Member Function Documentation

6.595.3.1 `virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket () throw (decaf::io::IOException) [virtual]`

Creates an unconnected **Socket** (p. 3281) object.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if the **Socket** (p. 3281) cannot be created.

Reimplemented from **decaf::net::SocketFactory** (p. 3302).

6.595.3.2 `virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (decaf::net::Socket * socket, std::string host, int port, bool autoClose) [virtual]`

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

- socket* The existing socket to layer over.
- host* The server host the original **Socket** (p. 3281) is connected to.
- port* The server port the original **Socket** (p. 3281) is connected to.
- autoClose* Should the layered over **Socket** (p. 3281) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 3281) instance that wraps the given **Socket** (p. 3281).

Exceptions

- IOException* if an I/O exception occurs while performing this operation.
- UnknownHostException* (p. 3649) if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3346).

6.595.3.3 virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (const decaf::net::InetAddress * *host*, int *port*) throw (decaf::io::IOException, decaf::net::UnknownHostException) [virtual]

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

- host* The host to connect the socket to.
- port* The port on the remote host to connect to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

- IOException* if an I/O error occurs while creating the **Socket** (p. 3281) object.
- UnknownHostException* (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3302).

6.595.3.4 virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (const std::string & *hostname*, int *port*) throw (decaf::io::IOException, decaf::net::UnknownHostException) [virtual]

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

- host* The host name or IP address to connect the socket to.

port The port on the remote host to connect to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3303).

```
6.595.3.5  virtual decaf::net::Socket* de-
            caf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (
            const std::string & name, int port, const decaf::net::InetAddress
            * ifAddress, int localPort ) throw ( decaf::io::IOException,
            decaf::net::UnknownHostException ) [virtual]
```

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host name or IP address to connect the socket to.

port The port on the remote host to connect to.

ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.

localPort The local port to bind the **Socket** (p. 3281) to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3303).

```
6.595.3.6  virtual decaf::net::Socket* de-
            caf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket
            ( const decaf::net::InetAddress * host, int port, const
            decaf::net::InetAddress * ifAddress, int localPort ) throw (
            decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]
```

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

The **Socket** (p. 3281) will be bound to the specified local address and port.

Parameters

host The host to connect the socket to.

port The port on the remote host to connect to.

ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.

localPort The local port to bind the **Socket** (p. 3281) to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3304).

6.595.3.7 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::getDefaultCipherSuites() [virtual]`

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

`getSupportedCipherSuites()` (p. 3347)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3347).

6.595.3.8 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::getSupportedCipherSuites() [virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

`getDefaultCipherSuites()` (p. 3347)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3347).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h`

6.596 decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream Class Reference

An output stream for reading data from an OpenSSL Socket instance.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream:

Public Member Functions

- **OpenSSLSocketInputStream** (**OpenSSLSocket** *socket)
- virtual **~OpenSSLSocketInputStream** ()
- virtual int **available** () const throw (decaf::io::IOException)

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

- virtual void **close** () throw (decaf::io::IOException)
- virtual long long **skip** (long long num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Not supported.

Protected Member Functions

- virtual int **doReadByte** () throw (io::IOException)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

6.596.1 Detailed Description

An output stream for reading data from an OpenSSL Socket instance.

Since

1.0

6.596.2 Constructor & Destructor Documentation

6.596.2.1 `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::OpenSSLSocketInputStream (OpenSSLSocket * socket)`

6.596.2.2 `virtual decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::~~OpenSSLSocketInputStream () [virtual]`

6.596.3 Member Function Documentation

6.596.3.1 `virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::available () const throw (decaf::io::IOException) [virtual]`

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented from `decaf::io::InputStream` (p. 1911).

6.596.3.2 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::close () throw (decaf::io::IOException) [virtual]`

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Closes the **InputStream** (p. 1909) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from `decaf::io::InputStream` (p. 1912).

6.596.3.3 `virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException) [protected, virtual]`

Reimplemented from `decaf::io::InputStream` (p. 1912).

6.596.3.4 `virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadByte () throw (io::IOException)` [protected, virtual]

Implements `decaf::io::InputStream` (p. 1912).

6.596.3.5 `virtual long long decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::skip (long long num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)` [virtual]

Not supported.

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of `InputStream` (p. 1909) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

Reimplemented from `decaf::io::InputStream` (p. 1917).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h`

6.597 decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream Class Reference

OutputStream implementation used to write data to an `OpenSSLSocket` (p. 2673) instance.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h>
```

Inheritance diagram for `decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream`:

Public Member Functions

- **OpenSSLSocketOutputStream** (**OpenSSLSocket** *socket)
- virtual **~OpenSSLSocketOutputStream** ()
- virtual void **close** () throw (decaf::io::IOException)

The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char c) throw (decaf::io::IOException)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

6.597.1 Detailed Description

OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2673) instance.

Since

1.0

6.597.2 Constructor & Destructor Documentation

6.597.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::OpenSSLSocketOutputStream (**OpenSSLSocket** * *socket*)

6.597.2.2 virtual
decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::~~OpenSSLSocketOutputStream () [virtual]

6.597.3 Member Function Documentation

6.597.3.1 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::close ()
throw (decaf::io::IOException) [virtual]

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2720).

6.597.3.2 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteArrayBounded (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2721).

6.597.3.3 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteByte (unsigned char c) throw (decaf::io::IOException)` [protected, virtual]

Implements `decaf::io::OutputStream` (p. 2721).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h`

6.598 `activemq::wireformat::openwire::OpenWireFormat` Class Reference

`#include <src/main/activemq/wireformat/openwire/OpenWireFormat.h>`

Inheritance diagram for `activemq::wireformat::openwire::OpenWireFormat`:

Public Member Functions

- `OpenWireFormat (const decaf::util::Properties &properties)`
*Constructs a new **OpenWireFormat** (p. 2700) object.*
- `virtual ~OpenWireFormat ()`
- `virtual bool hasNegotiator () const`
*Returns true if this **WireFormat** (p. 3712) has a Negotiator that needs to wrap the Transport that uses it.*
- `virtual Pointer< transport::Transport > createNegotiator (const Pointer< transport::Transport > &transport) throw (decaf::lang::exceptions::UnsupportedOperationException)`
If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.
- `void addMarshaller (marshal::DataStreamMarshaller *marshaller)`
Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.
- `virtual void marshal (const Pointer< commands::Command > &command, const activemq::transport::Transport *transport, decaf::io::DataOutputStream *out) throw (decaf::io::IOException)`
Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.
- `virtual Pointer< commands::Command > unmarshal (const activemq::transport::Transport *transport, decaf::io::DataInputStream *in) throw (decaf::io::IOException)`
Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form.

- virtual int **tightMarshalNestedObject1** (commands::DataStructure *object, utils::BooleanStream *bs) throw (decaf::io::IOException)

Utility method for Tight Marshaling the given object to the boolean stream passed.

- void **tightMarshalNestedObject2** (commands::DataStructure *o, decaf::io::DataOutputStream *ds, utils::BooleanStream *bs) throw (decaf::io::IOException)

Utility method that will Tight marshal some internally nested object that implements the DataStructure interface.

- commands::DataStructure * **tightUnmarshalNestedObject** (decaf::io::DataInputStream *dis, utils::BooleanStream *bs) throw (decaf::io::IOException)

Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.

- commands::DataStructure * **looseUnmarshalNestedObject** (decaf::io::DataInputStream *dis) throw (decaf::io::IOException)

Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.

- void **looseMarshalNestedObject** (commands::DataStructure *o, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Utility method to loosely Marshal an object that is derived from the DataStrucutre interface.

- void **renegotiateWireFormat** (const commands::WireFormatInfo &info) throw (decaf::lang::exceptions::IllegalStateException)

Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.

- virtual void **setPreferedWireFormatInfo** (const Pointer< commands::WireFormatInfo > &info) throw (decaf::lang::exceptions::IllegalStateException)

Configures this object using the provided WireformatInfo object.

- virtual const Pointer< commands::WireFormatInfo > & **getPreferedWireFormatInfo** () const

Gets the Preferred WireFormatInfo object that this class holds.

- bool **isStackTraceEnabled** () const

Checks if the stackTraceEnabled flag is on.

- void **setStackTraceEnabled** (bool stackTraceEnabled)

Sets if the stackTraceEnabled flag is on.

- bool **isTcpNoDelayEnabled** () const

Checks if the tcpNoDelayEnabled flag is on.

- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)

Sets if the tcpNoDelayEnabled flag is on.

- **int getVersion () const**
Get the current Wireformat Version.
- **void setVersion (int version) throw (decaf::lang::exceptions::IllegalArgumentException)**
Set the current Wireformat Version.
- **virtual bool inReceive () const**
Is there a Message being unmarshaled?
- **bool isCacheEnabled () const**
Checks if the cacheEnabled flag is on.
- **void setCacheEnabled (bool cacheEnabled)**
Sets if the cacheEnabled flag is on.
- **int getCacheSize () const**
Returns the currently set Cache size.
- **void setCacheSize (int value)**
Sets the current Cache size.
- **bool isTightEncodingEnabled () const**
Checks if the tightEncodingEnabled flag is on.
- **void setTightEncodingEnabled (bool tightEncodingEnabled)**
Sets if the tightEncodingEnabled flag is on.
- **bool isSizePrefixDisabled () const**
Checks if the sizePrefixDisabled flag is on.
- **void setSizePrefixDisabled (bool sizePrefixDisabled)**
Sets if the sizePrefixDisabled flag is on.
- **long long getMaxInactivityDuration () const**
Gets the MaxInactivityDuration setting.
- **void setMaxInactivityDuration (long long value)**
Sets the MaxInactivityDuration setting.
- **long long getMaxInactivityDurationInitialDelay () const**
Gets the MaxInactivityDurationInitialDelay setting.
- **void setMaxInactivityDurationInitialDelay (long long value)**
Sets the MaxInactivityDurationInitialDelay setting.

Protected Member Functions

- **commands::DataStructure * doUnmarshal** (decaf::io::DataInputStream *dis)
throw (decaf::io::IOException)
Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrcutre object once done, caller takes ownership of this object.
- **void destroyMarshallers** ()
Cleans up all registered Marshallers and empties the dataMarshallers vector.

Static Protected Attributes

- static const unsigned char **NULL__TYPE**
- static const int **DEFAULT__VERSION** = 1

6.598.1 Constructor & Destructor Documentation

6.598.1.1 **activemq::wireformat::openwire::OpenWireFormat::OpenWireFormat** (const decaf::util::Properties & *properties*)

Constructs a new **OpenWireFormat** (p. 2700) object.

Parameters

properties - can contain optional config params.

6.598.1.2 **virtual** **activemq::wireformat::openwire::OpenWireFormat::~~OpenWireFormat** () [virtual]

6.598.2 Member Function Documentation

6.598.2.1 **void activemq::wireformat::openwire::OpenWireFormat::addMarshaller** (marsh::DataStreamMarshaller * *marshaller*)

Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.

Parameters

marshaller - the Marshaler to add to the collection.

6.598.2.2 **virtual Pointer<transport::Transport> ac-** **tivemq::wireformat::openwire::OpenWireFormat::createNegotiator** (const Pointer< transport::Transport > & *transport*) throw (decaf::lang::exceptions::UnsupportedOperationException) [virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

Returns

new instance of a **WireFormatNegotiator** (p. 3751).

Implements **activemq::wireformat::WireFormat** (p. 3714).

6.598.2.3 void ac-
tivemq::wireformat::openwire::OpenWireFormat::destroyMarshallers ()
 [protected]

Cleans up all registered Marshallers and empties the dataMarshallers vector.

This should be called before a reconfiguration of the version marshallers, or on destruction of this object

6.598.2.4 commands::DataStructure* ac-
tivemq::wireformat::openwire::OpenWireFormat::doUnmarshal (
decaf::io::DataInputStream * *dis*) throw (decaf::io::IOException)
 [protected]

Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.

This method can return null if the type of the object to unmarshal is NULL, empty data.

Parameters

dis - DataInputStream to read from

Returns

new DataStructure* that the caller owns

Exceptions

IOException if an error occurs during the unmarshal

6.598.2.5 int activemq::wireformat::openwire::OpenWireFormat::getCacheSize ()
const [inline]

Returns the currently set Cache size.

Returns

the current value of the broker's cache size.

6.598.2.6 long long ac-
tivemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDuration
() const [inline]

Gets the MaxInactivityDuration setting.

Returns

maximum inactivity duration value in milliseconds.

6.598.2.7 `long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDurationInitialDelay () const [inline]`

Gets the MaxInactivityDurationInitialDelay setting.

Returns

maximum inactivity duration initial delay value in milliseconds.

6.598.2.8 `virtual const Pointer<commands::WireFormatInfo>& activemq::wireformat::openwire::OpenWireFormat::getPreferredWireFormatInfo () const [inline, virtual]`

Gets the Preferred WireFormatInfo object that this class holds.

Returns

pointer to a preferred WireFormatInfo object

6.598.2.9 `int activemq::wireformat::openwire::OpenWireFormat::getVersion () const [inline, virtual]`

Get the current Wireformat Version.

Returns

int that identifies the version

Implements **activemq::wireformat::WireFormat** (p. 3714).

6.598.2.10 `virtual bool activemq::wireformat::openwire::OpenWireFormat::hasNegotiator () const [inline, virtual]`

Returns true if this **WireFormat** (p. 3712) has a Negotiator that needs to wrap the Transport that uses it.

Returns

true if the **WireFormat** (p. 3712) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3714).

6.598.2.11 `virtual bool activemq::wireformat::openwire::OpenWireFormat::inReceive () const [inline, virtual]`

Is there a Message being unmarshaled?

Returns

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 3715).

6.598.2.12 `bool activemq::wireformat::openwire::OpenWireFormat::isCacheEnabled () const [inline]`

Checks if the cacheEnabled flag is on.

Returns

true if the flag is on.

6.598.2.13 `bool activemq::wireformat::openwire::OpenWireFormat::isSizePrefixDisabled () const [inline]`

Checks if the sizePrefixDisabled flag is on.

Returns

true if the flag is on.

6.598.2.14 `bool activemq::wireformat::openwire::OpenWireFormat::isStackTraceEnabled () const [inline]`

Checks if the stackTraceEnabled flag is on.

Returns

true if the flag is on.

6.598.2.15 `bool activemq::wireformat::openwire::OpenWireFormat::isTcpNoDelayEnabled () const [inline]`

Checks if the tcpNoDelayEnabled flag is on.

Returns

true if the flag is on.

6.598.2.16 `bool activemq::wireformat::openwire::OpenWireFormat::isTightEncodingEnabled () const [inline]`

Checks if the tightEncodingEnabled flag is on.

Returns

true if the flag is on.

6.598.2.17 `void activemq::wireformat::openwire::OpenWireFormat::looseMarshalNestedObject (commands::DataStructure * o, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)`

Utility method to loosely Marshal an object that is derived from the DataStrucutre interface.

The marshaled data is written to the passed in DataOutputStream.

Parameters

o - DataStructure derived Object to Marshal
dataOut - DataOutputStream to write the data to

Exceptions

IOException if an error occurs.

6.598.2.18 `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::looseUnmarshalNestedObject (decaf::io::DataInputStream * dis) throw (decaf::io::IOException)`

Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.

Will read the Data and construct a new DataStructure based Object, the pointer to the Object returned is now owned by the caller.

Parameters

dis - the DataInputStream to read the data from

Returns

a new DataStructure derived Object pointer

Exceptions

IOException if an error occurs.

6.598.2.19 `virtual void activemq::wireformat::openwire::OpenWireFormat::marshal (const Pointer< commands::Command > & command, const activemq::transport::Transport * transport, decaf::io::DataOutputStream * out) throw (decaf::io::IOException) [virtual]`

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters

command The Command to Marshal.
transport The Transport instance that called this method.
out The output stream to write the command to.

Exceptions

IOException

Implements `activemq::wireformat::WireFormat` (p. 3715).

6.598.2.20 `void activemq::wireformat::openwire::OpenWireFormat::renegotiateWireFormat (const commands::WireFormatInfo & info) throw (decaf::lang::exceptions::IllegalStateException)`

Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.

Parameters

info - The new Wireformat Info settings

Exceptions

IllegalStateException is wire format can't be negotiated.

6.598.2.21 `void activemq::wireformat::openwire::OpenWireFormat::setCacheEnabled (bool cacheEnabled) [inline]`

Sets if the cacheEnabled flag is on.

Parameters

cacheEnabled - true to turn flag is on

6.598.2.22 `void activemq::wireformat::openwire::OpenWireFormat::setCacheSize (int value) [inline]`

Sets the current Cache size.

Parameters

value - the value to send as the broker's cache size.

6.598.2.23 `void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDuration (long long value) [inline]`

Sets the MaxInactivityDuration setting.

Parameters

value - the Max inactivity duration value in milliseconds.

6.598.2.24 void ac-
tivismq::wireformat::openwire::OpenWireFormat::setMaxInactivityDurationInitialDelay
(long long *value*) [inline]

Sets the MaxInactivityDurationInitialDelay setting.

Parameters

value - the Max inactivity Initial Delay duration value in milliseconds.

6.598.2.25 virtual void ac-
tivismq::wireformat::openwire::OpenWireFormat::setPreferredWireFormatInfo
(const Pointer< commands::WireFormatInfo > & *info*) throw (
decaf::lang::exceptions::IllegalStateException) [virtual]

Configures this object using the provided WireFormatInfo object.

Parameters

info - a WireFormatInfo object, takes ownership.

6.598.2.26 void ac-
tivismq::wireformat::openwire::OpenWireFormat::setSizePrefixDisabled
(bool *sizePrefixDisabled*) [inline]

Sets if the sizePrefixDisabled flag is on.

Parameters

sizePrefixDisabled - true to turn flag is on

6.598.2.27 void ac-
tivismq::wireformat::openwire::OpenWireFormat::setStackTraceEnabled
(bool *stackTraceEnabled*) [inline]

Sets if the stackTraceEnabled flag is on.

Parameters

stackTraceEnabled - true to turn flag is on

6.598.2.28 void ac-
tivismq::wireformat::openwire::OpenWireFormat::setTcpNoDelayEnabled
(bool *tcpNoDelayEnabled*) [inline]

Sets if the tcpNoDelayEnabled flag is on.

Parameters

tcpNoDelayEnabled - true to turn flag is on

6.598.2.29 `void activemq::wireformat::openwire::OpenWireFormat::setTightEncodingEnabled (bool tightEncodingEnabled) [inline]`

Sets if the `tightEncodingEnabled` flag is on.

Parameters

tightEncodingEnabled - true to turn flag is on

6.598.2.30 `void activemq::wireformat::openwire::OpenWireFormat::setVersion (int version) throw (decaf::lang::exceptions::IllegalArgumentException) [virtual]`

Set the current Wireformat Version.

Parameters

version - int that identifies the version

Implements `activemq::wireformat::WireFormat` (p. 3715).

6.598.2.31 `virtual int activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject1 (commands::DataStructure * object, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Utility method for Tight Marshaling the given object to the boolean stream passed.

Parameters

object - The DataStructure to marshal

bs - the BooleanStream to write to

Returns

size of the data returned.

6.598.2.32 `void activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject2 (commands::DataStructure * o, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) throw (decaf::io::IOException)`

Utility method that will Tight marshal some internally nested object that implements the DataStructure interface.

Writes the data to the Data Output Stream provided.

Parameters

o - DataStructure object

ds - DataOutputSteam for writing

bs - BooleanStream

Exceptions

IOException if an error occurs.

6.598.2.33 `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::tightUnmarshalNestedObject (decaf::io::DataInputStream * dis, utils::BooleanStream * bs) throw (decaf::io::IOException)`

Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.

The DataStructure instance that is returned is now the property of the caller.

Parameters

dis - DataInputStream to read from

bs - BooleanStream to read from

Returns

Newly allocated DataStructure Object

Exceptions

IOException if an error occurs.

6.598.2.34 `virtual Pointer<commands::Command> activemq::wireformat::openwire::OpenWireFormat::unmarshal (const activemq::transport::Transport * transport, decaf::io::DataInputStream * in) throw (decaf::io::IOException) [virtual]`

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form.

Returns a Pointer to the newly un-marshaled Command.

Parameters

transport - Pointer to the transport that is making this request.

in - the input stream to read the command from.

Returns

the newly marshaled Command, caller owns the pointer

Exceptions

IOException

Implements `activemq::wireformat::WireFormat` (p. 3716).

6.598.3 Field Documentation

6.598.3.1 `const int`
`activemq::wireformat::openwire::OpenWireFormat::DEFAULT_`
`VERSION = 1` [static, protected]

6.598.3.2 `const unsigned char`
`activemq::wireformat::openwire::OpenWireFormat::NULL_`
`TYPE` [static, protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormat.h`

6.599 `activemq::wireformat::openwire::OpenWireFormatFactory` Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h>
```

Inheritance diagram for `activemq::wireformat::openwire::OpenWireFormatFactory`:

Public Member Functions

- `OpenWireFormatFactory ()`

Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.

- `virtual ~OpenWireFormatFactory ()`
- `virtual Pointer< wireformat::WireFormat > createWireFormat (const decaf::util::Properties &properties) throw (decaf::lang::exceptions::IllegalStateException)`

*Creates a new **WireFormat** (p. 3712) Object passing it a set of properties from which it can obtain any optional settings.*

6.599.1 Constructor & Destructor Documentation

6.599.1.1 `activemq::wireformat::openwire::OpenWireFormatFactory::OpenWireFormatFactory ()` [inline]

Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.

URL options ----- `wireFormat.stackTraceEnabled` `wireFormat.cacheEnabled` `wireFormat.tcpNoDelayEnabled` `wireFormat.tightEncodingEnabled` `wireFormat.sizePrefixDisabled` `wireFormat.maxInactivityDuration` `wireFormat.maxInactivityDurationInitialDelay`

6.599.1.2 virtual
activemq::wireformat::openwire::OpenWireFormatFactory::~OpenWireFormatFactory
() [inline, virtual]

6.599.2 Member Function Documentation

6.599.2.1 virtual Pointer<wireformat::WireFormat> activemq::wireformat::openwire::OpenWireFormatFactory::createWireFormat (const decaf::util::Properties & *properties*) throw (decaf::lang::exceptions::IllegalStateException) [virtual]

Creates a new **WireFormat** (p. 3712) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

properties - the Properties for this **WireFormat** (p. 3712)

Implements **activemq::wireformat::WireFormatFactory** (p. 3717).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h

6.600 activemq::wireformat::openwire::OpenWireFormatNegotiator Class Reference

#include <src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h>

Inheritance diagram for activemq::wireformat::openwire::OpenWireFormatNegotiator:

Public Member Functions

- **OpenWireFormatNegotiator** (**OpenWireFormat** *wireFormat, const **Pointer**< **transport::Transport** > &next)

Constructor - Initializes this object around another Transport.

- virtual ~**OpenWireFormatNegotiator** ()
- virtual void **oneway** (const **Pointer**< **commands::Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Sends a one-way command.

- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Sends the given request to the server and waits for the response.

- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Sends the given request to the server and waits for the response.

- virtual void **onCommand** (const **Pointer**< **commands::Command** > &command)

This is called in the context of the nested transport's reading thread.

- virtual void **onTransportException** (transport::Transport *source, const **decaf::lang::Exception** &ex)

Event handler for an exception from a command transport.

- virtual void **start** () throw (decaf::io::IOException)

Starts this transport object and creates the thread for polling on the input stream for commands.

- virtual void **close** () throw (decaf::io::IOException)

Stops the polling thread and closes the streams.

6.600.1 Constructor & Destructor Documentation

6.600.1.1 activemq::wireformat::openwire::OpenWireFormatNegotiator::OpenWireFormatNegotiator (OpenWireFormat * *wireFormat*, const **Pointer**< transport::Transport > & *next*)

Constructor - Initializes this object around another Transport.

Parameters

wireFormat - The **WireFormat** (p.3712) object we use to negotiate

next - The next transport in the chain

6.600.1.2 virtual activemq::wireformat::openwire::OpenWireFormatNegotiator::~~OpenWireFormatNegotiator () [virtual]

6.600.2 Member Function Documentation

6.600.2.1 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::close () throw (decaf::io::IOException) [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if errors occur.

Reimplemented from **activemq::transport::TransportFilter** (p. 3638).

6.600.2.2 `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onCommand (const Pointer< commands::Command > & command) [virtual]`

This is called in the context of the nested transport's reading thread.

In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

Parameters

command the received from the nested transport.

Reimplemented from `activemq::transport::TransportFilter` (p. 3640).

6.600.2.3 `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::oneway (const Pointer< commands::Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from `activemq::transport::TransportFilter` (p. 3640).

6.600.2.4 `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onTransportException (transport::Transport * source, const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters

source The source of the exception

ex The exception.

6.600.2.5 `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given request to the server and waits for the response.

First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters

command The request to send.

Returns

the response from the server.

Exceptions

IOException if an error occurs with the request.

Reimplemented from `activemq::transport::TransportFilter` (p. 3642).

6.600.2.6 `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given request to the server and waits for the response.

First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters

command The request to send.

timeout The time to wait for the response.

Returns

the response from the server.

Exceptions

IOException if an error occurs with the request.

Reimplemented from `activemq::transport::TransportFilter` (p. 3641).

6.600.2.7 `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::start () throw (decaf::io::IOException) [virtual]`

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions

- IOException*** if an error occurs or if this transport has already been closed.

Reimplemented from **activemq::transport::TransportFilter** (p. 3642).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireFormatNegotiator.h**

6.601 activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h>
```

Inheritance diagram for **activemq::wireformat::openwire::OpenWireResponseBuilder**:

Public Member Functions

- OpenWireResponseBuilder** ()
- virtual **~OpenWireResponseBuilder** ()
- virtual **Pointer< commands::Response > buildResponse** (const **Pointer< commands::Command > &command**)

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

- virtual void **buildIncomingCommands** (const **Pointer< commands::Command > &command**, **decaf::util::StlQueue< Pointer< commands::Command > > &queue**)

When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

6.601.1 Constructor & Destructor Documentation

- 6.601.1.1** `activemq::wireformat::openwire::OpenWireResponseBuilder::OpenWireResponseBuilder () [inline]`
- 6.601.1.2** `virtual activemq::wireformat::openwire::OpenWireResponseBuilder::~~OpenWireResponseBuilder () [inline, virtual]`

6.601.2 Member Function Documentation

- 6.601.2.1** `virtual void activemq::wireformat::openwire::OpenWireResponseBuilder::buildIncomingCommands (const Pointer< commands::Command > & command, decaf::util::StlQueue< Pointer< commands::Command > > & queue) [virtual]`

When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

Parameters

- command* - The Command being sent to the Broker.
- queue* - Queue of Command sent back from the broker.

Implements `activemq::transport::mock::ResponseBuilder` (p. 3080).

- 6.601.2.2** `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireResponseBuilder::buildResponse (const Pointer< commands::Command > & command) [virtual]`

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

Parameters

- command* - The command to build a response for

Returns

- A Response object pointer, or NULL if no response.

Implements `activemq::transport::mock::ResponseBuilder` (p. 3081).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h`

6.602 decaf::io::OutputStream Class Reference

Base interface for any class that wants to represent an output stream of bytes.

`#include <src/main/decaf/io/OutputStream.h>`

Inheritance diagram for decaf::io::OutputStream:

Public Member Functions

- **OutputStream** ()
- virtual **~OutputStream** ()
- virtual void **close** () throw (decaf::io::IOException)

- virtual void **flush** () throw (decaf::io::IOException)

- virtual void **write** (unsigned char c) throw (decaf::io::IOException)
Writes a single byte to the output stream.
- virtual void **write** (const unsigned char *buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Writes an array of bytes to the output stream in order starting at buffer[offset] and proceeding until the number of bytes specified by the length argument are written or an error occurs.
- virtual std::string **toString** () const
Output a String representation of this object.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)=0 throw (decaf::io::IOException)
- virtual void **doWriteArray** (const unsigned char *buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

6.602.1 Detailed Description

Base interface for any class that wants to represent an output stream of bytes.

Since

1.0

6.602.2 Constructor & Destructor Documentation

6.602.2.1 decaf::io::OutputStream::OutputStream ()

6.602.2.2 virtual decaf::io::OutputStream::~~OutputStream () [virtual]

6.602.3 Member Function Documentation

6.602.3.1 virtual void decaf::io::OutputStream::close () throw (decaf::io::IOException) [virtual]

The default implementation of this method does nothing.

Reimplemented in **decaf::internal::io::StandardErrorOutputStream** (p. 3352), **decaf::internal::io::StandardOutputStream** (p. 3354), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 2699), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 3510), **decaf::io::FilterOutputStream** (p. 1778), and **decaf::util::zip::DeflaterOutputStream** (p. 1607).

6.602.3.2 virtual void decaf::io::OutputStream::doWriteArray (const unsigned char * *buffer*, int *size*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [protected, virtual]

Reimplemented in decaf::io::BufferedOutputStream (p. 868), and decaf::io::FilterOutputStream (p. 1779).

6.602.3.3 virtual void decaf::io::OutputStream::doWriteArrayBounded (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [protected, virtual]

Reimplemented in activemq::io::LoggingOutputStream (p. 2252), decaf::internal::io::StandardErrorOutputStream (p. 3352), decaf::internal::io::StandardOutputStream (p. 3355), decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream (p. 2699), decaf::internal::net::tcp::TcpSocketOutputStream (p. 3510), decaf::io::BufferedOutputStream (p. 868), decaf::io::ByteArrayOutputStream (p. 952), decaf::io::DataOutputStream (p. 1475), decaf::io::FilterOutputStream (p. 1779), decaf::util::zip::CheckedOutputStream (p. 1059), and decaf::util::zip::DeflaterOutputStream (p. 1607).

6.602.3.4 virtual void decaf::io::OutputStream::doWriteByte (unsigned char *value*) throw (decaf::io::IOException) [protected, pure virtual]

Implemented in activemq::io::LoggingOutputStream (p. 2252), decaf::internal::io::StandardErrorOutputStream (p. 3352), decaf::internal::io::StandardOutputStream (p. 3355), decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream (p. 2700), decaf::internal::net::tcp::TcpSocketOutputStream (p. 3510), decaf::io::BufferedOutputStream (p. 868), decaf::io::ByteArrayOutputStream (p. 952), decaf::io::DataOutputStream (p. 1475), decaf::io::FilterOutputStream (p. 1779), decaf::util::zip::CheckedOutputStream (p. 1059), and decaf::util::zip::DeflaterOutputStream (p. 1607).

6.602.3.5 virtual void decaf::io::OutputStream::flush () throw (decaf::io::IOException) [virtual]

The default implementation of this method does nothing.

Reimplemented in decaf::internal::io::StandardErrorOutputStream (p. 3352), decaf::internal::io::StandardOutputStream (p. 3355), decaf::io::BufferedOutputStream (p. 868), and decaf::io::FilterOutputStream (p. 1779).

6.602.3.6 virtual void decaf::io::OutputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3463).

6.602.3.7 `virtual void decaf::io::OutputStream::notify ()
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3464).

6.602.3.8 `virtual void decaf::io::OutputStream::notifyAll ()
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3465).

6.602.3.9 `virtual std::string decaf::io::OutputStream::toString () const
[virtual]`

Output a String representation of this object.

The default version of this method just prints the Class Name.

Returns

a string representation of the object.

Reimplemented in `decaf::io::ByteArrayOutputStream` (p. 953), and `decaf::io::FilterOutputStream` (p. 1780).

6.602.3.10 `virtual bool decaf::io::OutputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3466).

6.602.3.11 `virtual void decaf::io::OutputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3467).

6.602.3.12 `virtual void decaf::io::OutputStream::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3468).

6.602.3.13 `virtual void decaf::io::OutputStream::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3470).

6.602.3.14 `virtual void decaf::io::OutputStream::wait (long long millisecs,
int nanos) throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3471).

6.602.3.15 `virtual void decaf::io::OutputStream::write (unsigned char c) throw
(decaf::io::IOException) [virtual]`

Writes a single byte to the output stream.

The default implementation of this method calls the pure virtual method doWriteByte which must be implemented by any subclass of the **OutputStream** (p. 2718).

Parameters

c The byte to write to the sink.

Exceptions

IOException (p. 2003) if an I/O error occurs.

6.602.3.16 `virtual void decaf::io::OutputStream::write (const unsigned char * buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes an array of bytes to the output stream.

The entire contents of the given vector are written to the output stream.

The default implementation of this method simply calls the `doWriteArray` which writes the contents of the array using the `doWriteByte` method repeatedly. It is recommended that a subclass override `doWriteArray` to provide more performant means of writing the array.

Parameters

buffer The vector of bytes to write.

size The size of the buffer passed.

Exceptions

IOException (p. 2003) if an I/O error occurs.

NullPointerException thrown if buffer is Null.

IndexOutOfBoundsException if size value is negative.

6.602.3.17 `virtual void decaf::io::OutputStream::write (const unsigned char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes an array of bytes to the output stream in order starting at `buffer[offset]` and proceeding until the number of bytes specified by the `length` argument are written or an error occurs.

The default implementation of this method simply calls the `doWriteArrayBounded` method which writes the contents of the array using the `doWriteByte` method repeatedly. It is recommended that a subclass override `doWriteArrayBounded` to provide more performant means of writing the array.

Parameters

buffer The array of bytes to write.

size The size of the buffer array passed.

offset The position to start writing in buffer.

length The number of bytes from the buffer to be written.

Exceptions

IOException (p. 2003) if an I/O error occurs.

NullPointerException thrown if buffer is Null.

IndexOutOfBoundsException if the offset + length > size. or one of the parameters is negative.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**OutputStream.h**

6.603 decaf::io::OutputStreamWriter Class Reference

A class for turning a character stream into a byte stream.

#include <src/main/decaf/io/OutputStreamWriter.h>

Inheritance diagram for decaf::io::OutputStreamWriter:

Public Member Functions

- **OutputStreamWriter** (**OutputStream** *stream, bool own=false)
*Creates a new **OutputStreamWriter** (p. 2726).*
- virtual ~**OutputStreamWriter** ()
- virtual void **close** () throw (decaf::io::IOException)
- virtual void **flush** () throw (decaf::io::IOException)

Protected Member Functions

- virtual void **doWriteArrayBounded** (const char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Override this method to customize the functionality of the method write(char buffer, int size, int offset, int length).*
- virtual void **checkClosed** () const throw (decaf::io::IOException)

6.603.1 Detailed Description

A class for turning a character stream into a byte stream.

See also

InputStreamReader (p. 1919)

Since

1.0

6.603.2 Constructor & Destructor Documentation

6.603.2.1 decaf::io::OutputStreamWriter::OutputStreamWriter (*OutputStream* * *stream*, *bool own* = *false*)

Creates a new **OutputStreamWriter** (p. 2726).

Parameters

stream The **OutputStream** (p. 2718) to wrap. (cannot be NULL).

own Indicates whether this instance own the given **OutputStream** (p. 2718). If true then the **OutputStream** (p. 2718) is destroyed when this class is.

Exceptions

NullPointerException if the stream is NULL.

6.603.2.2 virtual decaf::io::OutputStreamWriter::~~OutputStreamWriter () [virtual]

6.603.3 Member Function Documentation

6.603.3.1 virtual void decaf::io::OutputStreamWriter::checkClosed () const throw (decaf::io::IOException) [protected, virtual]

6.603.3.2 virtual void decaf::io::OutputStreamWriter::close () throw (decaf::io::IOException) [virtual]

6.603.3.3 virtual void decaf::io::OutputStreamWriter::doWriteArrayBounded (const char * *buffer*, *int size*, *int offset*, *int length*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [protected, virtual]

Override this method to customize the functionality of the method write(char* buffer, int size, int offset, int length).

All subclasses must override this method to provide the basic **Writer** (p. 3756) functionality.

Implements **decaf::io::Writer** (p. 3759).

6.603.3.4 virtual void decaf::io::OutputStreamWriter::flush () throw (decaf::io::IOException) [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/io/OutputStreamWriter.h

6.604 activemq::commands::PartialCommand Class Reference

```
#include <src/main/activemq/commands/PartialCommand.h>
```

Inheritance diagram for `activemq::commands::PartialCommand`:

Public Member Functions

- **PartialCommand** ()
- virtual **~PartialCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **PartialCommand * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual int **getCommandId** () const
- virtual void **setCommandId** (int commandId)
- virtual const std::vector< unsigned char > & **getData** () const
- virtual std::vector< unsigned char > & **getData** ()
- virtual void **setData** (const std::vector< unsigned char > &data)

Static Public Attributes

- static const unsigned char **ID_PARTIALCOMMAND** = 60

Protected Attributes

- int **commandId**
- std::vector< unsigned char > **data**

6.604.1 Constructor & Destructor Documentation

6.604.1.1 `activemq::commands::PartialCommand::PartialCommand ()`

6.604.1.2 `virtual activemq::commands::PartialCommand::~~PartialCommand ()`
[virtual]

6.604.2 Member Function Documentation

6.604.2.1 `virtual PartialCommand* activemq::commands::PartialCommand::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

Reimplemented in `activemq::commands::LastPartialCommand` (p. 2157).

6.604.2.2 `virtual void activemq::commands::PartialCommand::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1555).

Reimplemented in `activemq::commands::LastPartialCommand` (p. 2157).

6.604.2.3 `virtual bool activemq::commands::PartialCommand::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1556).

Reimplemented in `activemq::commands::LastPartialCommand` (p. 2157).

6.604.2.4 `virtual int activemq::commands::PartialCommand::getCommandId ()`
`const [virtual]`

6.604.2.5 `virtual std::vector<unsigned char>& ac-`
`tivemq::commands::PartialCommand::getData ()`
`[virtual]`

6.604.2.6 `virtual const std::vector<unsigned char>& ac-`
`tivemq::commands::PartialCommand::getData () const`
`[virtual]`

6.604.2.7 `virtual unsigned char ac-`
`tivemq::commands::PartialCommand::getDataStructureType () const`
`[virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2157).

6.604.2.8 `virtual void activemq::commands::PartialCommand::setCommandId (`
`int commandId) [virtual]`

6.604.2.9 `virtual void activemq::commands::PartialCommand::setData (const`
`std::vector< unsigned char > & data) [virtual]`

6.604.2.10 `virtual std::string activemq::commands::PartialCommand::toString ()`
`const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 767).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2158).

6.604.3 Field Documentation

- 6.604.3.1 `int activemq::commands::PartialCommand::commandId` [protected]
- 6.604.3.2 `std::vector<unsigned char> activemq::commands::PartialCommand::data`
[protected]
- 6.604.3.3 `const unsigned char activemq::commands::PartialCommand::ID_ -
PARTIALCOMMAND = 60` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/PartialCommand.h`

6.605 activemq::wireformat::openwire::marshal::v6::PartialCommandMa Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2731).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller`:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **com-
mands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn,
utils::BooleanStream *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **com-
mands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**de-
caf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **com-
mands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut,
utils::BooleanStream *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.605.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2731). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.605.2 Constructor & Destructor Documentation

6.605.2.1 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::PartialCommandMarshaller () [inline]

6.605.2.2 virtual activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::~~PartialCommandMarshaller () [inline, virtual]

6.605.3 Member Function Documentation

6.605.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2159).

6.605.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2160).

6.605.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1518).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2160).

6.605.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2160).

```

6.605.3.5  virtual int ac-
            tivemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, utils::BooleanStream * bs ) throw (
            decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller` (p. 2161).

```

6.605.3.6  virtual void ac-
            tivemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller` (p. 2161).

6.605.3.7 virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2162).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h

6.606 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2735).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.606.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2735). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.606.2 Constructor & Destructor Documentation

6.606.2.1 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::PartialCommandMarshaller () [inline]

6.606.2.2 virtual
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::~~PartialCommandMarshaller () [inline, virtual]

6.606.3 Member Function Documentation

6.606.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2167).

6.606.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2167).

6.606.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1518).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2168).

6.606.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 2168).

```
6.606.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 2169).

```
6.606.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2169).

6.606.3.7 virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2169).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h`

6.607 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2739).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller**:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()

- virtual **commands::DataStructure * createObject ()** const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType ()** const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)** throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)** throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)** throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn)** throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut)** throw (decaf::io::IOException)

Write a object instance to data output stream.

6.607.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2739). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.607.2 Constructor & Destructor Documentation

6.607.2.1 `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::PartialCommandMarshaller () [inline]`

6.607.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::~~PartialCommandMarshaller () [inline, virtual]`

6.607.3 Member Function Documentation

6.607.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2171).

6.607.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2171).

6.607.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2172).

6.607.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2172).

6.607.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2173).


```

6.607.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2173).

```

6.607.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2173).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h

6.608 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2744).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.608.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p.2744). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.608.2 Constructor & Destructor Documentation

6.608.2.1 **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::PartialCommandMarshaller**
() [inline]

6.608.2.2 **virtual**
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::~~PartialCommandMarshaller
() [inline, virtual]

6.608.3 Member Function Documentation

6.608.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p.2163).

6.608.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::getDataStructureType**
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p.2164).

6.608.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseMarshal**
(**OpenWireFormat * wireFormat**, **commands::DataStructure * dataStructure**, **decaf::io::DataOutputStream * dataOut**) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 2164).

6.608.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 2164).

6.608.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2165).

6.608.3.6 virtual void **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal2**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2165).

6.608.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2165).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h`

6.609 **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2748).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.609.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2748). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.609.2 Constructor & Destructor Documentation

6.609.2.1 **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::PartialCommandMarshaller ()** [inline]

6.609.2.2 **virtual activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::~~PartialCommandMarshaller ()** [inline, virtual]

6.609.3 Member Function Documentation

6.609.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::createObject () const** [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2175).

6.609.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::getDataStructureType () const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2175).

6.609.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 2176).

6.609.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 2176).

6.609.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2177).

```
6.609.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2177).

```
6.609.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2177).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h`

6.610 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2752).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.610.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p.2752). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.610.2 Constructor & Destructor Documentation

6.610.2.1 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::PartialCommandMarshaller () [inline]

6.610.2.2 virtual activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::~~PartialCommandMarshaller () [inline, virtual]

6.610.3 Member Function Documentation

6.610.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p.2179).

6.610.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2179).

6.610.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2180).

6.610.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2180).

6.610.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2181).

6.610.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2181).

```

6.610.3.7 virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2181).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h`

6.611 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 2756) that is a template on a Type and is **Thread** (p. 3520) Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/Pointer.h>
```

Public Types

- `typedef T * PointerType`
- `typedef T & ReferenceType`
- `typedef REFCOUNTER CounterType`

Public Member Functions

- **Pointer** ()
Default Constructor.
- **Pointer** (const **PointerType** value)
*Explicit Constructor, creates a **Pointer** (p. 2756) that contains value with a single reference.*
- **Pointer** (const **Pointer** &value) throw ()

Copy constructor.

- template<typename T1 , typename R1 >
Pointer (const **Pointer**< T1, R1 > &value) throw ()
Copy constructor.
- template<typename T1 , typename R1 >
Pointer (const **Pointer**< T1, R1 > &value, const **STATIC_CAST_TOKEN** &) throw ()
Static Cast constructor.
- template<typename T1 , typename R1 >
Pointer (const **Pointer**< T1, R1 > &value, const **DYNAMIC_CAST_TOKEN** &) throw (decaf::lang::exceptions::ClassCastException)
Dynamic Cast constructor.
- virtual ~**Pointer** () throw ()
- void **reset** (T *value)
*Resets the **Pointer** (p. 2756) to hold the new value.*
- T * **release** ()
*Releases the **Pointer** (p. 2756) held and resets the internal pointer value to Null.*
- **PointerType** **get** () const
*Gets the real pointer that is contained within this **Pointer** (p. 2756).*
- void **swap** (**Pointer** &value) throw ()
***Exception** (p. 1712) Safe Swap Function.*
- **Pointer** & **operator=** (const **Pointer** &right) throw ()
*Assigns the value of right to this **Pointer** (p. 2756) and increments the reference Count.*
- template<typename T1 , typename R1 >
Pointer & **operator=** (const **Pointer**< T1, R1 > &right) throw ()
- **ReferenceType** **operator*** ()
Dereference Operator, returns a reference to the Contained value.
- **ReferenceType** **operator*** () const
- **PointerType** **operator->** ()
Indirection Operator, returns a pointer to the Contained value.
- **PointerType** **operator->** () const
- bool **operator!** () const
- template<typename T1 , typename R1 >
bool **operator==** (const **Pointer**< T1, R1 > &right) const
- template<typename T1 , typename R1 >
bool **operator!=** (const **Pointer**< T1, R1 > &right) const
- template<typename T1 >
Pointer< T1, **CounterType** > **dynamicCast** () const
- template<typename T1 >
Pointer< T1, **CounterType** > **staticCast** () const

Friends

- `bool operator==(const Pointer &left, const T *right)`
- `bool operator==(const T *left, const Pointer &right)`
- `bool operator!=(const Pointer &left, const T *right)`
- `bool operator!=(const T *left, const Pointer &right)`

6.611.1 Detailed Description

```
template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> class decaf::lang::Pointer<
T, REFCOUNTER >
```

Decaf's implementation of a Smart **Pointer** (p. 2756) that is a template on a Type and is **Thread** (p. 3520) Safe if the default Reference Counter is used. This **Pointer** (p. 2756) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of **ReferenceCounter**.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 2756) instances in the same manner as normal pointer, except that it does not provide an overload of operators (<, <=, >, >=). To allow use of a **Pointer** (p. 2756) in a STL container that requires it, **Pointer** (p. 2756) provides an implementation of `std::less`.

Since

1.0

6.611.2 Member Typedef Documentation

6.611.2.1 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> typedef
REFCOUNTER decaf::lang::Pointer< T, REFCOUNTER
>::CounterType`

6.611.2.2 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> typedef T*
decaf::lang::Pointer< T, REFCOUNTER >::PointerType`

6.611.2.3 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> typedef T&
decaf::lang::Pointer< T, REFCOUNTER >::ReferenceType`

6.611.3 Constructor & Destructor Documentation

6.611.3.1 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> decaf::lang::Pointer<
T, REFCOUNTER >::Pointer () [inline]`

Default Constructor.

Initialized the contained pointer to NULL, using the `->` operator results in an exception unless reset to contain a real value.

Referenced by `decaf::lang::Pointer< TransactionId >::reset()`.

6.611.3.2 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::Pointer<
T, REFCOUNTER >::Pointer (const PointerType value) [inline,
explicit]`

Explicit Constructor, creates a **Pointer** (p.2756) that contains value with a single reference.
This object now has ownership until a call to release.

Parameters

value - instance of the type we are containing here.

6.611.3.3 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::Pointer<
T, REFCOUNTER >::Pointer (const Pointer< T, REFCOUNTER > &
value) throw () [inline]`

Copy constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter.

6.611.3.4 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename
T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer
(const Pointer< T1, R1 > & value) throw () [inline]`

Copy constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter.

6.611.3.5 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename
T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer
(const Pointer< T1, R1 > & value, const STATIC_CAST_TOKEN
&) throw () [inline]`

Static Cast constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter performing a static cast on the value contained in the source **Pointer** (p.2756) object.

Parameters

value - **Pointer** (p.2756) instance to cast to this type.

6.611.3.6 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> template<typename
T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer
(const Pointer< T1, R1 > & value, const DYNAMIC_CAST_TOKEN
&) throw (decaf::lang::exceptions::ClassCastException) [inline]`

Dynamic Cast constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter performing a dynamic cast on the value contained in the source **Pointer** (p. 2756) object. If the cast fails and return NULL then this method throws a `ClassCastException`.

Parameters

value - **Pointer** (p. 2756) instance to cast to this type.

Exceptions

ClassCastException if the dynamic cast returns NULL

6.611.3.7 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> virtual
decaf::lang::Pointer< T, REFCOUNTER >::~~Pointer () throw ()
[inline, virtual]`

6.611.4 Member Function Documentation

6.611.4.1 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> template<typename
T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T,
REFCOUNTER >::dynamicCast () const [inline]`

6.611.4.2 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> PointerType
decaf::lang::Pointer< T, REFCOUNTER >::get () const [inline]`

Gets the real pointer that is contained within this **Pointer** (p. 2756).

This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p. 2756). Use at your own risk.

Returns

the contained pointer.

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::equals()`, `activemq::state::ConnectionState::getTransactionState()`, `decaf::lang::operator!=()`, `decaf::lang::Pointer< TransactionId >::operator!=()`, `std::less< decaf::lang::Pointer< T > >::operator()`, `decaf::lang::operator==()`, `decaf::lang::Pointer< TransactionId >::operator==()`, and `activemq::state::ConnectionState::removeTempDestination()`.

- 6.611.4.3** `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> bool
decaf::lang::Pointer< T, REFCOUNTER >::operator! () const
[inline]`
- 6.611.4.4** `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename
T1 , typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER
>::operator!= (const Pointer< T1, R1 > & right) const [inline]`
- 6.611.4.5** `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> ReferenceType
decaf::lang::Pointer< T, REFCOUNTER >::operator* () const
[inline]`
- 6.611.4.6** `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> ReferenceType
decaf::lang::Pointer< T, REFCOUNTER >::operator* () [inline]`

Dereference Operator, returns a reference to the Contained value.

This method throws an `NullPointerException` if the contained value is `NULL`.

Returns

reference to the contained pointer.

Exceptions

NullPointerException if the contained value is `Null`

- 6.611.4.7** `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> PointerType
decaf::lang::Pointer< T, REFCOUNTER >::operator-> () [inline]`

Indirection Operator, returns a pointer to the Contained value.

This method throws an `NullPointerException` if the contained value is `NULL`.

Returns

reference to the contained pointer.

Exceptions

NullPointerException if the contained value is `Null`

- 6.611.4.8 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> PointerType
decaf::lang::Pointer< T, REFCOUNTER >::operator-> () const
[inline]`
- 6.611.4.9 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> Pointer&
decaf::lang::Pointer< T, REFCOUNTER >::operator= (const Pointer<
T, REFCOUNTER > & right) throw () [inline]`

Assigns the value of *right* to this **Pointer** (p. 2756) and increments the reference Count.

Parameters

right - **Pointer** (p. 2756) on the right hand side of an operator= call to this.

- 6.611.4.10 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename
T1 , typename R1 > Pointer& decaf::lang::Pointer< T, REFCOUNTER
>::operator= (const Pointer< T1, R1 > & right) throw () [inline]`
- 6.611.4.11 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename
T1 , typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER
>::operator== (const Pointer< T1, R1 > & right) const [inline]`
- 6.611.4.12 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> T*
decaf::lang::Pointer< T, REFCOUNTER >::release () [inline]`

Releases the **Pointer** (p. 2756) held and resets the internal pointer value to Null.

This method is not guaranteed to be safe if the **Pointer** (p. 2756) is held by more than one object or this method is called from more than one thread.

Parameters

value - The new value to contain.

Returns

The pointer instance that was held by this **Pointer** (p. 2756) object, the pointer is no longer owned by this **Pointer** (p. 2756) and won't be freed when this **Pointer** (p. 2756) goes out of scope.

Referenced by `decaf::lang::Pointer< TransactionId >::Pointer()`.

- 6.611.4.13 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCounter> void
decaf::lang::Pointer< T, REFCOUNTER >::reset (T * value)
[inline]`

Resets the **Pointer** (p. 2756) to hold the new value.

Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p.2756) to a NULL pointer.

Parameters

value - The new value to contain.

6.611.4.14 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> template<typename
T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T,
REFCOUNTER >::staticCast () const [inline]`

6.611.4.15 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> void
decaf::lang::Pointer< T, REFCOUNTER >::swap (Pointer< T,
REFCOUNTER > & value) throw () [inline]`

Exception (p.1712) Safe Swap Function.

Parameters

value - the value to swap with this.

Referenced by decaf::lang::Pointer< TransactionId >::operator=(), and decaf::lang::Pointer< TransactionId >::swap().

6.611.5 Friends And Related Function Documentation

6.611.5.1 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> bool operator!= (
const Pointer< T, REFCOUNTER > & left, const T * right)
[friend]`

6.611.5.2 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> bool operator!= (
const T * left, const Pointer< T, REFCOUNTER > & right)
[friend]`

6.611.5.3 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> bool operator== (
const Pointer< T, REFCOUNTER > & left, const T * right)
[friend]`

6.611.5.4 `template<typename T, typename REFCOUNTER =
decaf::util::concurrent::atomic::AtomicRefCount> bool operator== (
const T * left, const Pointer< T, REFCOUNTER > & right)
[friend]`

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.612 decaf::lang::PointerComparator< T, R > Class Template Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2756) instance.

```
#include <src/main/decaf/lang/Pointer.h>
```

Inheritance diagram for decaf::lang::PointerComparator< T, R >:

Public Member Functions

- virtual bool **operator()** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const
- virtual int **compare** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const

6.612.1 Detailed Description

```
template<typename T, typename R = decaf::util::concurrent::atomic::AtomicRefCounter>
class decaf::lang::PointerComparator< T, R >
```

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2756) instance. This can be useful in the case where a series of values in a Collection is more efficiently accessed in the Objects Natural Order and not the underlying pointers location in memory.

Also this allows **Pointer** (p. 2756) objects that Point to different instances of the same type to be compared based on the comparison of the object itself and not just the value of the pointer.

6.612.2 Member Function Documentation

6.612.2.1 `template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual int decaf::lang::PointerComparator< T, R >::compare (const Pointer< T, R > & left, const Pointer< T, R > & right) const [inline, virtual]`

6.612.2.2 `template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual bool decaf::lang::PointerComparator< T, R >::operator() (const Pointer< T, R > & left, const Pointer< T, R > & right) const [inline, virtual]`

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.613 activemq::cmsutil::PooledSession Class Reference

A pooled session object that wraps around a delegate session.

```
#include <src/main/activemq/cmsutil/PooledSession.h>
```

Inheritance diagram for activemq::cmsutil::PooledSession:

Public Member Functions

- **PooledSession** (**SessionPool** *pool, **cms::Session** *session)
- virtual **~PooledSession** ()
Does nothing.
- virtual **cms::Session** * **getSession** ()
Returns a non-constant reference to the internal session object.
- virtual const **cms::Session** * **getSession** () const
Returns a constant reference to the internal session object.
- virtual void **close** () throw (cms::CMSEException)
Returns this session back to the pool, but does not close or destroy the internal session object.
- virtual void **commit** () throw (cms::CMSEException)
Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** () throw (cms::CMSEException)
Rolls back all messages done in this transaction and releases any locks currently held.
- virtual void **recover** () throw (cms::CMSEException)
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual **cms::MessageConsumer** * **createDurableConsumer** (const **cms::Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false) throw (cms::CMSEException)

Creates a durable subscriber to the specified topic, using a Message selector.

- virtual **cms::MessageConsumer** * **createCachedConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal) throw (cms::CMSEException)

First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.

- virtual **cms::MessageProducer** * **createProducer** (const **cms::Destination** *destination) throw (cms::CMSEException)

Creates a MessageProducer to send messages to the specified destination.

- virtual **cms::MessageProducer** * **createCachedProducer** (const **cms::Destination** *destination) throw (cms::CMSEException)

First checks the internal producer cache and creates one if none exist for the given destination.

- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue) throw (cms::CMSEException)

Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue, const std::string &selector) throw (cms::CMSEException)

Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::Queue** * **createQueue** (const std::string &queueName) throw (cms::CMSEException)

Creates a queue identity given a Queue name.

- virtual **cms::Topic** * **createTopic** (const std::string &topicName) throw (cms::CMSEException)

Creates a topic identity given a Queue name.

- virtual **cms::TemporaryQueue** * **createTemporaryQueue** () throw (cms::CMSEException)

Creates a TemporaryQueue object.

- virtual **cms::TemporaryTopic** * **createTemporaryTopic** () throw (cms::CMSEException)

Creates a TemporaryTopic object.

- virtual **cms::Message** * **createMessage** () throw (cms::CMSEException)

Creates a new Message.

- virtual **cms::BytesMessage** * **createBytesMessage** () throw (cms::CMSEException)

Creates a BytesMessage.

- virtual **cms::BytesMessage** * **createBytesMessage** (const unsigned char *bytes, int bytesSize) throw (cms::CMSEException)

Creates a BytesMessage and sets the payload to the passed value.

- virtual **cms::StreamMessage** * **createStreamMessage** () throw (cms::CMSEException)

Creates a new StreamMessage.

- virtual **cms::TextMessage * createTextMessage** () throw (cms::CMSEException)

Creates a new TextMessage.

- virtual **cms::TextMessage * createTextMessage** (const std::string &text) throw (cms::CMSEException)

Creates a new TextMessage and set the text to the value given.

- virtual **cms::MapMessage * createMapMessage** () throw (cms::CMSEException)

Creates a new MapMessage.

- virtual **cms::Session::AcknowledgeMode getAcknowledgeMode** () const throw (cms::CMSEException)

Returns the acknowledgment mode of the session.

- virtual bool **isTransacted** () const throw (cms::CMSEException)

Gets if the Sessions is a Transacted Session.

- virtual void **unsubscribe** (const std::string &name) throw (cms::CMSEException)

Unsubscribes a durable subscription that has been created by a client.

Protected Member Functions

- **PooledSession** (const **PooledSession** &)
- **PooledSession & operator=** (const **PooledSession** &)

6.613.1 Detailed Description

A pooled session object that wraps around a delegate session. Calls to close this session only result in giving the session back to the pool.

6.613.2 Constructor & Destructor Documentation

6.613.2.1 **activemq::cmsutil::PooledSession::PooledSession** (const **PooledSession** &) [inline, protected]

6.613.2.2 **activemq::cmsutil::PooledSession::PooledSession** (SessionPool * *pool*, cms::Session * *session*)

6.613.2.3 virtual **activemq::cmsutil::PooledSession::~~PooledSession** () [virtual]

Does nothing.

6.613.3 Member Function Documentation

6.613.3.1 `virtual void activemq::cmsutil::PooledSession::close () throw (cms::CMSEException) [virtual]`

Returns this session back to the pool, but does not close or destroy the internal session object.
Implements `cms::Session` (p. 3152).

6.613.3.2 `virtual void activemq::cmsutil::PooledSession::commit () throw (cms::CMSEException) [inline, virtual]`

Commits all messages done in this transaction and releases any locks currently held.

Exceptions

CMSEException - If an internal error occurs.

IllegalStateException - if the method is not called by a transacted session.

Implements `cms::Session` (p. 3152).

References `cms::Session::commit()`.

6.613.3.3 `virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * queue, const std::string & selector) throw (cms::CMSEException) [virtual]`

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

queue the Queue to browse

selector the Message selector to filter which messages are browsed.

Returns

New QueueBrowser that is owned by the caller.

Exceptions

CMSEException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements `cms::Session` (p. 3153).

6.613.3.4 `virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * queue) throw (cms::CMSEException) [virtual]`

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

queue the Queue to browse

Returns

New QueueBrowser that is owned by the caller.

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements **cms::Session** (p. 3152).

6.613.3.5 `virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage () throw (cms::CMSException) [inline, virtual]`

Creates a BytesMessage.

Exceptions

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 3153).

6.613.3.6 `virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage (const unsigned char * bytes, int bytesSize) throw (cms::CMSException) [inline, virtual]`

Creates a BytesMessage and sets the payload to the passed value.

Parameters

bytes an array of bytes to set in the message

bytesSize the size of the bytes array, or number of bytes to use

Exceptions

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 3153).

6.613.3.7 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createCachedConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal) throw (cms::CMSException) [virtual]`

First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.

If created, the consumer is added to the pool's lifecycle manager.

Parameters

destination the destination to receive on
selector the selector to use
noLocal whether or not to receive messages from the same connection

Returns

the consumer resource

Exceptions

cms::CMSEException (p. 1074) if something goes wrong.

6.613.3.8 `virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createCachedProducer (const cms::Destination * destination) throw (cms::CMSEException)`
 [virtual]

First checks the internal producer cache and creates one if none exist for the given destination.
 If created, the producer is added to the pool's lifecycle manager.

Parameters

destination the destination to send on

Returns

the producer resource

Exceptions

cms::CMSEException (p. 1074) if something goes wrong.

6.613.3.9 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * destination) throw (cms::CMSEException)`
 [inline, virtual]

Creates a MessageConsumer for the specified destination.

Parameters

destination the Destination that this consumer receiving messages for.

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

CMSEException - If an internal error occurs.
InvalidDestinationException - if an invalid destination is specified.

Implements **cms::Session** (p. 3154).

6.613.3.10 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * destination, const std::string & selector) throw (cms::CMSException) [inline, virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

destination the Destination that this consumer receiving messages for.

selector the Message Selector to use

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements **cms::Session** (p. 3154).

6.613.3.11 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal) throw (cms::CMSException) [inline, virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

destination the Destination that this consumer receiving messages for.

selector the Message Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements **cms::Session** (p. 3155).

6.613.3.12 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createDurableConsumer (const cms::Topic * destination, const std::string & name, const std::string & selector, bool noLocal = false) throw (cms::CMSEException) [inline, virtual]`

Creates a durable subscriber to the specified topic, using a Message selector.

Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

Parameters

destination the topic to subscribe to

name The name used to identify the subscription

selector the Message Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new durable MessageConsumer that is owned by the caller (caller deletes)

Exceptions

CMSEException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements `cms::Session` (p. 3155).

6.613.3.13 `virtual cms::MapMessage* activemq::cmsutil::PooledSession::createMapMessage () throw (cms::CMSEException) [inline, virtual]`

Creates a new MapMessage.

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::Session` (p. 3156).

6.613.3.14 `virtual cms::Message* activemq::cmsutil::PooledSession::createMessage () throw (cms::CMSEException) [inline, virtual]`

Creates a new Message.

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::Session` (p. 3156).

6.613.3.15 `virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createProducer (const cms::Destination * destination) throw (cms::CMSException)`
[inline, virtual]

Creates a MessageProducer to send messages to the specified destination.

Parameters

destination the Destination to send on

Returns

New MessageProducer that is owned by the caller.

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

Implements **cms::Session** (p. 3156).

6.613.3.16 `virtual cms::Queue* activemq::cmsutil::PooledSession::createQueue (const std::string & queueName) throw (cms::CMSException)`
[inline, virtual]

Creates a queue identity given a Queue name.

Parameters

queueName the name of the new Queue

Returns

new Queue pointer that is owned by the caller.

Exceptions

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 3156).

6.613.3.17 `virtual cms::StreamMessage* activemq::cmsutil::PooledSession::createStreamMessage () throw (cms::CMSException)` [inline, virtual]

Creates a new StreamMessage.

Exceptions

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 3157).

6.613.3.18 `virtual cms::TemporaryQueue* activemq::cmsutil::PooledSession::createTemporaryQueue () throw (cms::CMSEException) [inline, virtual]`

Creates a TemporaryQueue object.

Returns

new TemporaryQueue pointer that is owned by the caller.

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::Session` (p. 3157).

6.613.3.19 `virtual cms::TemporaryTopic* activemq::cmsutil::PooledSession::createTemporaryTopic () throw (cms::CMSEException) [inline, virtual]`

Creates a TemporaryTopic object.

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::Session` (p. 3157).

6.613.3.20 `virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage (const std::string & text) throw (cms::CMSEException) [inline, virtual]`

Creates a new TextMessage and set the text to the value given.

Parameters

text the initial text for the message

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::Session` (p. 3158).

6.613.3.21 `virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage () throw (cms::CMSEException) [inline, virtual]`

Creates a new TextMessage.

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::Session` (p. 3158).

6.613.3.22 `virtual cms::Topic* activemq::cmsutil::PooledSession::createTopic
(const std::string & topicName) throw (cms::CMSEException)
[inline, virtual]`

Creates a topic identity given a Queue name.

Parameters

topicName the name of the new Topic

Returns

new Topic pointer that is owned by the caller.

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 3158).

6.613.3.23 `virtual cms::Session::AcknowledgeMode ac-
tivemq::cmsutil::PooledSession::getAcknowledgeMode () const throw
(cms::CMSEException) [inline, virtual]`

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 3158).

6.613.3.24 `virtual cms::Session* activemq::cmsutil::PooledSession::getSession ()
[inline, virtual]`

Returns a non-constant reference to the internal session object.

Returns

the session object.

6.613.3.25 `virtual const cms::Session* activemq::cmsutil::PooledSession::getSession
() const [inline, virtual]`

Returns a constant reference to the internal session object.

Returns

the session object.

6.613.3.26 `virtual bool activemq::cmsutil::PooledSession::isTransacted () const throw (cms::CMSEException) [inline, virtual]`

Gets if the Sessions is a Transacted Session.

Returns

transacted true - false.

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::Session` (p. 3159).

6.613.3.27 `PooledSession& activemq::cmsutil::PooledSession::operator= (const PooledSession &) [inline, protected]`

6.613.3.28 `virtual void activemq::cmsutil::PooledSession::recover () throw (cms::CMSEException) [inline, virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions

CMSEException - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException - if the method is called by a transacted session.

Implements `cms::Session` (p. 3159).

6.613.3.29 `virtual void activemq::cmsutil::PooledSession::rollback () throw (cms::CMSEException) [inline, virtual]`

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions

CMSEException - If an internal error occurs.

IllegalStateException - if the method is not called by a transacted session.

Implements `cms::Session` (p. 3159).

6.613.3.30 `virtual void activemq::cmsutil::PooledSession::unsubscribe (const std::string & name) throw (cms::CMSException) [inline, virtual]`

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters

name The name used to identify this subscription

Exceptions

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 3160).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**PooledSession.h**

6.614 decaf::util::concurrent::PooledThread Class Reference

```
#include <src/main/decaf/util/concurrent/PooledThread.h>
```

Inheritance diagram for decaf::util::concurrent::PooledThread:

Public Member Functions

- **PooledThread** (**ThreadPool** *pool)
Constructor.
- virtual **~PooledThread** ()
- virtual void **run** ()
*Run Method for this object waits for something to be enqueued on the **ThreadPool** (p. 3531) and then grabs it and calls its run method.*
- virtual void **stop** () throw (lang::Exception)
Stops the Thread, thread will complete its task if currently running one, and then die.
- virtual bool **isBusy** ()
Checks to see if the thread is busy, if busy it means that this thread has taken a task from the ThreadPool's queue and is processing it.
- virtual void **setPooledThreadListener** (**PooledThreadListener** *listener)
*Adds a listener to this **PooledThread** (p. 2777) to be notified when this thread starts and completes a task.*

- virtual **PooledThreadListener** * **getPooledThreadListener** ()

*Removes a listener for this **PooledThread** (p. 2777) to be notified when this thread starts and completes a task.*

6.614.1 Constructor & Destructor Documentation

6.614.1.1 **decaf::util::concurrent::PooledThread::PooledThread** (**ThreadPool** * *pool*)

Constructor.

Parameters

pool the parant **ThreadPool** (p. 3531) object

6.614.1.2 **virtual decaf::util::concurrent::PooledThread::~~PooledThread** () [virtual]

6.614.2 Member Function Documentation

6.614.2.1 **virtual PooledThreadListener* decaf::util::concurrent::PooledThread::getPooledThreadListener** () [inline, virtual]

Removes a listener for this **PooledThread** (p. 2777) to be notified when this thread starts and completes a task.

Returns

a pointer to this thread's listener or NULL

6.614.2.2 **virtual bool decaf::util::concurrent::PooledThread::isBusy** () [inline, virtual]

Checks to see if the thread is busy, if busy it means that this thread has taken a task from the **ThreadPool**'s queue and is processing it.

Returns

true if the Thread is busy

6.614.2.3 **virtual void decaf::util::concurrent::PooledThread::run** () [virtual]

Run Method for this object waits for something to be enqueued on the **ThreadPool** (p. 3531) and then grabs it and calls its run method.

Reimplemented from **decaf::lang::Thread** (p. 3527).

6.614.2.4 virtual void decaf::util::concurrent::PooledThread::setPooledThreadListener (PooledThreadListener * *listener*) [inline, virtual]

Adds a listener to this PooledThread (p. 2777) to be notified when this thread starts and completes a task.

Parameters

listener the listener to send notifications to.

6.614.2.5 virtual void decaf::util::concurrent::PooledThread::stop () throw (lang::Exception) [virtual]

Stops the Thread, thread will complete its task if currently running one, and then die.

Does not block.

Exceptions

Exception

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**PooledThread.h**

6.615 decaf::util::concurrent::PooledThreadListener Class Reference

Abstract Listener Interface for users of ThreadPool (p. 3531).

```
#include <src/main/decaf/util/concurrent/PooledThreadListener.h>
```

Inheritance diagram for decaf::util::concurrent::PooledThreadListener:

Public Member Functions

- virtual ~PooledThreadListener ()
- virtual void onTaskStarted (PooledThread *thread)=0
Called by a pooled thread when it is about to begin executing a new task.
- virtual void onTaskCompleted (PooledThread *thread)=0
Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.
- virtual void onTaskException (PooledThread *thread, lang::Exception &ex)=0
*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2777) is now no longer running.*

6.615.1 Detailed Description

Abstract Listener Interface for users of `ThreadPool` (p. 3531). The implementor of this class receives events related to the execution and termination of threads running in the `ThreadPool` (p. 3531).

Since

1.0

6.615.2 Constructor & Destructor Documentation

6.615.2.1 `virtual`
`decaf::util::concurrent::PooledThreadListener::~~PooledThreadListener (`
`)` [inline, virtual]

6.615.3 Member Function Documentation

6.615.3.1 `virtual void` `de-`
`caf::util::concurrent::PooledThreadListener::onTaskCompleted (`
`PooledThread * thread)` [pure virtual]

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.

Parameters

thread - Pointer the the Pooled Thread that is making this call.

Implemented in `decaf::util::concurrent::ThreadPool` (p. 3535).

6.615.3.2 `virtual void` `de-`
`caf::util::concurrent::PooledThreadListener::onTaskException (`
`PooledThread * thread, lang::Exception & ex)` [pure virtual]

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the `PooledThread` (p. 2777) is now no longer running.

Parameters

thread - Pointer to the Pooled Thread that is making this call

ex - The Exception that occurred.

Implemented in `decaf::util::concurrent::ThreadPool` (p. 3535).

6.615.3.3 `virtual void` `decaf::util::concurrent::PooledThreadListener::onTaskStarted`
`(PooledThread * thread)` [pure virtual]

Called by a pooled thread when it is about to begin executing a new task.

Parameters

thread - Pointer to the Pooled Thread that is making this call

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3535).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/PooledThreadListener.h`

6.616 decaf::net::PortUnreachableException Class Reference

```
#include <src/main/decaf/net/PortUnreachableException.h>
```

Inheritance diagram for `decaf::net::PortUnreachableException`:

Public Member Functions

- **PortUnreachableException** () throw ()
Default Constructor.
- **PortUnreachableException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **PortUnreachableException** (const **PortUnreachableException** &ex) throw ()
Copy Constructor.
- **PortUnreachableException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **PortUnreachableException** (const std::exception *cause) throw ()
Constructor.
- **PortUnreachableException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **PortUnreachableException** * clone () const
Clones this exception.
- virtual ~**PortUnreachableException** () throw ()

6.616.1 Constructor & Destructor Documentation

6.616.1.1 `decaf::net::PortUnreachableException::PortUnreachableException () throw () [inline]`

Default Constructor.

6.616.1.2 `decaf::net::PortUnreachableException::PortUnreachableException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.616.1.3 `decaf::net::PortUnreachableException::PortUnreachableException (const PortUnreachableException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.616.1.4 `decaf::net::PortUnreachableException::PortUnreachableException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.616.1.5 `decaf::net::PortUnreachableException::PortUnreachableException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.616.1.6 `decaf::net::PortUnreachableException::PortUnreachableException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.616.1.7 virtual
decaf::net::PortUnreachableException::~~PortUnreachableException ()
throw () [inline, virtual]

6.616.2 Member Function Documentation

6.616.2.1 virtual PortUnreachableException* decaf::net::PortUnreachableException::clone () const
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

- a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3300).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/PortUnreachableException.h`

6.617 activemq::core::PrefetchPolicy Class Reference

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

```
#include <src/main/activemq/core/PrefetchPolicy.h>
```

Inheritance diagram for `activemq::core::PrefetchPolicy`:

Public Member Functions

- virtual **~PrefetchPolicy ()**
- virtual void **setDurableTopicPrefetch (int value)=0**
Sets the amount of prefetched messages for a Durable Topic.
- virtual int **getDurableTopicPrefetch () const =0**
Gets the amount of messages to prefetch for a Durable Topic.

- virtual void **setQueuePrefetch** (int value)=0
Sets the amount of prefetched messages for a Queue.
- virtual int **getQueuePrefetch** () const =0
Gets the amount of messages to prefetch for a Queue.
- virtual void **setQueueBrowserPrefetch** (int value)=0
Sets the amount of prefetched messages for a Queue Browser.
- virtual int **getQueueBrowserPrefetch** () const =0
Gets the amount of messages to prefetch for a Queue Browser.
- virtual void **setTopicPrefetch** (int value)=0
Sets the amount of prefetched messages for a Topic.
- virtual int **getTopicPrefetch** () const =0
Gets the amount of messages to prefetch for a Topic.
- virtual int **getMaxPrefetchLimit** (int value) const =0
Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.
- virtual **PrefetchPolicy** * **clone** () const =0
Clone the Policy and return a new pointer to that clone.
- virtual void **configure** (const **decaf::util::Properties** &properties)
Checks the supplied properties object for properties matching the configurable settings of this class.

Protected Member Functions

- **PrefetchPolicy** ()

6.617.1 Detailed Description

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

Since

3.2.0

6.617.2 Constructor & Destructor Documentation

6.617.2.1 `activemq::core::PrefetchPolicy::PrefetchPolicy ()` [protected]

6.617.2.2 `virtual activemq::core::PrefetchPolicy::~~PrefetchPolicy ()` [virtual]

6.617.3 Member Function Documentation

6.617.3.1 `virtual PrefetchPolicy* activemq::core::PrefetchPolicy::clone () const`
[pure virtual]

Clone the Policy and return a new pointer to that clone.

Returns

pointer to a new **PrefetchPolicy** (p. 2783) instance that is a clone of this one.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1567).

6.617.3.2 `virtual void activemq::core::PrefetchPolicy::configure (const decaf::util::Properties & properties)` [virtual]

Checks the supplied properties object for properties matching the configurable settings of this class.

The default implementation looks for properties named with the prefix `cms.PrefetchPolicy.XXX` where XXX is the name of a property with a public setter method. For instance `cms.PrefetchPolicy.topicPrefetch` will be used to set the value of the topic prefetch limit.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

Parameters

properties The Properties object used to configure this object.

Exceptions

NumberFormatException if a property that is numeric cannot be converted

IllegalArgumentException if a property can't be converted to the correct type.

6.617.3.3 `virtual int activemq::core::PrefetchPolicy::getDurableTopicPrefetch () const`
[pure virtual]

Gets the amount of messages to prefetch for a Durable Topic.

Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1567).

6.617.3.4 `virtual int activemq::core::PrefetchPolicy::getMaxPrefetchLimit (int value) const` [pure virtual]

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

Returns

the allowable value for a prefetch limit, either requested or the max.

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1567).

6.617.3.5 `virtual int activemq::core::PrefetchPolicy::getQueueBrowserPrefetch () const` [pure virtual]

Gets the amount of messages to prefetch for a Queue Browser.

Returns

value of the number of messages to prefetch.

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1567).

6.617.3.6 `virtual int activemq::core::PrefetchPolicy::getQueuePrefetch () const` [pure virtual]

Gets the amount of messages to prefetch for a Queue.

Returns

value of the number of messages to prefetch.

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1568).

6.617.3.7 `virtual int activemq::core::PrefetchPolicy::getTopicPrefetch () const` [pure virtual]

Gets the amount of messages to prefetch for a Topic.

Returns

value of the number of messages to prefetch.

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1568).

6.617.3.8 `virtual void activemq::core::PrefetchPolicy::setDurableTopicPrefetch (int value)` [pure virtual]

Sets the amount of prefetched messages for a Durable Topic.

Parameters

value The number of messages to prefetch.

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1568).

6.617.3.9 virtual void activemq::core::PrefetchPolicy::setQueueBrowserPrefetch (int *value*) [pure virtual]

Sets the amount of prefetched messages for a Queue Browser.

Parameters

value The number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1568).

6.617.3.10 virtual void activemq::core::PrefetchPolicy::setQueuePrefetch (int *value*) [pure virtual]

Sets the amount of prefetched messages for a Queue.

Parameters

value The number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1569).

6.617.3.11 virtual void activemq::core::PrefetchPolicy::setTopicPrefetch (int *value*) [pure virtual]

Sets the amount of prefetched messages for a Topic.

Parameters

value The number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1569).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/PrefetchPolicy.h`

6.618 activemq::util::PrimitiveList Class Reference

List of primitives.

```
#include <src/main/activemq/util/PrimitiveList.h>
```

Inheritance diagram for `activemq::util::PrimitiveList`:

Public Member Functions

- **PrimitiveList ()**
Default Constructor, creates an Empty list.

- virtual `~PrimitiveList ()`
- **PrimitiveList** (const `decaf::util::List< PrimitiveValueNode > &src`)
Copy Constructor.
- **PrimitiveList** (const **PrimitiveList** &src)
Copy Constructor.
- `std::string toString () const`
Converts the contents into a formatted string that can be output in a Log File or other debugging tool.
- virtual `bool getBool (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)`
Gets the Boolean value at the specified index.
- virtual `void setBool (std::size_t index, bool value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual `unsigned char getByte (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)`
Gets the Byte value at the specified index.
- virtual `void setByte (std::size_t index, unsigned char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual `char getChar (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)`
Gets the Character value at the specified index.
- virtual `void setChar (std::size_t index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual `short getShort (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)`
Gets the Short value at the specified index.
- virtual `void setShort (std::size_t index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual int **getInt** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Integer value at the specified index.

- virtual void **setInt** (std::size_t index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual long long **getLong** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Long value at the specified index.

- virtual void **setLong** (std::size_t index, long long value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual float **getFloat** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Float value at the specified index.

- virtual void **setFloat** (std::size_t index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual double **getDouble** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Double value at the specified index.

- virtual void **setDouble** (std::size_t index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual std::string **getString** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the String value at the specified index.

- virtual void **setString** (std::size_t index, const std::string &value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual std::vector< unsigned char > **getByteArray** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Byte Array value at the specified index.

- virtual void **setByteArray** (std::size_t index, const std::vector< unsigned char > &value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

6.618.1 Detailed Description

List of primitives.

6.618.2 Constructor & Destructor Documentation

6.618.2.1 activemq::util::PrimitiveList::PrimitiveList ()

Default Constructor, creates an Empty list.

6.618.2.2 virtual activemq::util::PrimitiveList::~~PrimitiveList () [virtual]

6.618.2.3 activemq::util::PrimitiveList::PrimitiveList (const decaf::util::List< PrimitiveValueNode > & src)

Copy Constructor.

Parameters

src - the Decaf List of PrimitiveNodeValues to copy

6.618.2.4 activemq::util::PrimitiveList::PrimitiveList (const PrimitiveList & src)

Copy Constructor.

Parameters

src - the **PrimitiveList** (p. 2787) to copy

6.618.3 Member Function Documentation

6.618.3.1 `virtual bool activemq::util::PrimitiveList::getBool (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Boolean value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > `size()` (p. 3369)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.618.3.2 `virtual unsigned char activemq::util::PrimitiveList::getBytes (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Byte value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > `size()` (p. 3369)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.618.3.3 `virtual std::vector<unsigned char> activemq::util::PrimitiveList::getBytesArray (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Byte Array value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is $> \text{size}()$ (p. 3369)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.618.3.4 `virtual char activemq::util::PrimitiveList::getChar (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Character value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is $> \text{size}()$ (p. 3369)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.618.3.5 `virtual double activemq::util::PrimitiveList::getDouble (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Double value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is $> \text{size}()$ (p. 3369)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.618.3.6 virtual float activemq::util::PrimitiveList::getFloat (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Float value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > size() (p. 3369)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.618.3.7 virtual int activemq::util::PrimitiveList::getInt (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Integer value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > size() (p. 3369)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.618.3.8 virtual long long activemq::util::PrimitiveList::getLong (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Long value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is $> \text{size}()$ (p. 3369)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.618.3.9 `virtual short activemq::util::PrimitiveList::getShort (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Short value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is $> \text{size}()$ (p. 3369)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.618.3.10 `virtual std::string activemq::util::PrimitiveList::getString (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the String value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is $> \text{size}()$ (p. 3369)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.618.3.11 `virtual void activemq::util::PrimitiveList::setBool (std::size_t index, bool value) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if `index > size()` (p. 3369).

6.618.3.12 `virtual void activemq::util::PrimitiveList::setByte (std::size_t index, unsigned char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if `index > size()` (p. 3369).

6.618.3.13 `virtual void activemq::util::PrimitiveList::setByteArray (std::size_t index, const std::vector< unsigned char > & value) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if `index > size()` (p. 3369).

6.618.3.14 `virtual void activemq::util::PrimitiveList::setChar (std::size_t index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if `index > size()` (p. 3369).

6.618.3.15 `virtual void activemq::util::PrimitiveList::setDouble
(std::size_t index, double value) throw (
decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if `index > size()` (p. 3369).

6.618.3.16 `virtual void activemq::util::PrimitiveList::setFloat
(std::size_t index, float value) throw (
decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if `index > size()` (p. 3369).

6.618.3.17 `virtual void activemq::util::PrimitiveList::setInt (std::size_t index, int
value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
[virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if index > size() (p. 3369).

6.618.3.18 virtual void activemq::util::PrimitiveList::setLong
(std::size_t index, long long value) throw (
decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if index > size() (p. 3369).

6.618.3.19 virtual void activemq::util::PrimitiveList::setShort
(std::size_t index, short value) throw (
decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if index > size() (p. 3369).

6.618.3.20 virtual void activemq::util::PrimitiveList::setString (
std::size_t index, const std::string & value) throw (
decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if index > size() (p. 3369).

6.618.3.21 std::string activemq::util::PrimitiveList::toString () const

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

Returns

formatted String of all elements in the list.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveList.h**

6.619 activemq::util::PrimitiveMap Class Reference

Map of named primitives.

```
#include <src/main/activemq/util/PrimitiveMap.h>
```

Inheritance diagram for activemq::util::PrimitiveMap:

Public Member Functions

- **PrimitiveMap ()**
Default Constructor, creates an empty map.
- virtual **~PrimitiveMap ()**
- **PrimitiveMap (const decaf::util::Map< std::string, PrimitiveValueNode > &source)**
Copy Constructor.
- **PrimitiveMap (const PrimitiveMap &source)**
Copy Constructor.
- std::string **toString () const**
Converts the contents into a formatted string that can be output in a Log File or other debugging tool.
- virtual bool **getBool (const std::string &key) const**
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setBool** (const std::string &key, bool value)

Sets the value at key to the specified type.

- virtual unsigned char **getByte** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setByte** (const std::string &key, unsigned char value)

Sets the value at key to the specified type.

- virtual char **getChar** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setChar** (const std::string &key, char value)

Sets the value at key to the specified type.

- virtual short **getShort** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setShort** (const std::string &key, short value)

Sets the value at key to the specified type.

- virtual int **getInt** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setInt** (const std::string &key, int value)

Sets the value at key to the specified type.

- virtual long long **getLong** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setLong** (const std::string &key, long long value)

Sets the value at key to the specified type.

- virtual float **getFloat** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, de-
cafe::lang::exceptions::UnsupportedOperationException)

Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setFloat** (const std::string &key, float value)

Sets the value at key to the specified type.

- virtual double **getDouble** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, de-
cafe::lang::exceptions::UnsupportedOperationException)

Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setDouble** (const std::string &key, double value)

Sets the value at key to the specified type.

- virtual std::string **getString** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, de-
cafe::lang::exceptions::UnsupportedOperationException)

Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setString** (const std::string &key, const std::string &value)

Sets the value at key to the specified type.

- virtual std::vector< unsigned char > **getByteArray** (const std::string
&key) const throw (decaf::lang::exceptions::NoSuchElementException, de-
cafe::lang::exceptions::UnsupportedOperationException)

Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setByteArray** (const std::string &key, const std::vector< unsigned char >
&value)

Sets the value at key to the specified type.

6.619.1 Detailed Description

Map of named primitives.

6.619.2 Constructor & Destructor Documentation

6.619.2.1 activemq::util::PrimitiveMap::PrimitiveMap ()

Default Constructor, creates an empty map.

6.619.2.2 virtual `activemq::util::PrimitiveMap::~~PrimitiveMap ()` [virtual]

6.619.2.3 `activemq::util::PrimitiveMap::PrimitiveMap (const decaf::util::Map< std::string, PrimitiveValueNode > & source)`

Copy Constructor.

Parameters

source The Decaf Library Map instance whose elements will be copied into this Map.

6.619.2.4 `activemq::util::PrimitiveMap::PrimitiveMap (const PrimitiveMap & source)`

Copy Constructor.

Parameters

source The **PrimitiveMap** (p.2798) whose elements will be copied into this Map.

6.619.3 Member Function Documentation

6.619.3.1 virtual `bool activemq::util::PrimitiveMap::getBool (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)` [virtual]

Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

key - the location to return the value from.

Returns

the value at key in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.619.3.2 virtual `unsigned char activemq::util::PrimitiveMap::getByte (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)` [virtual]

Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

key - the location to return the value from.

Returns

the value at *key* in the type requested.

Exceptions

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

```
6.619.3.3 virtual std::vector<unsigned char> activemq::util::PrimitiveMap::getByteArray ( const std::string & key ) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]
```

Gets the Byte Array value at the given *key*, if the *key* is not in the map or cannot be returned as the requested value then an exception of type *NoSuchElementException* is thrown.

Parameters

key - the location to return the value from.

Returns

the value at *key* in the type requested.

Exceptions

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

```
6.619.3.4 virtual char activemq::util::PrimitiveMap::getChar ( const std::string & key ) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]
```

Gets the Character value at the given *key*, if the *key* is not in the map or cannot be returned as the requested value then an exception of type *NoSuchElementException* is thrown.

Parameters

key - the location to return the value from.

Returns

the value at *key* in the type requested.

Exceptions

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.619.3.5 **virtual double** **activemq::util::PrimitiveMap::getDouble** (**const** **std::string** & *key*) **const throw** (**decaf::lang::exceptions::NoSuchElementException**, **decaf::lang::exceptions::UnsupportedOperationException**) [virtual]

Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

key - the location to return the value from.

Returns

the value at key in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.619.3.6 **virtual float** **activemq::util::PrimitiveMap::getFloat** (**const** **std::string** & *key*) **const throw** (**decaf::lang::exceptions::NoSuchElementException**, **decaf::lang::exceptions::UnsupportedOperationException**) [virtual]

Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

key - the location to return the value from.

Returns

the value at key in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.619.3.7 **virtual int** **activemq::util::PrimitiveMap::getInt** (**const** **std::string** & *key*) **const throw** (**decaf::lang::exceptions::NoSuchElementException**, **decaf::lang::exceptions::UnsupportedOperationException**) [virtual]

Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

key - the location to return the value from.

Returns

the value at *key* in the type requested.

Exceptions

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.619.3.8 `virtual long long activemq::util::PrimitiveMap::getLong (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Long value at the given *key*, if the *key* is not in the map or cannot be returned as the requested value then an exception of type *NoSuchElementException* is thrown.

Parameters

key - the location to return the value from.

Returns

the value at *key* in the type requested.

Exceptions

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.619.3.9 `virtual short activemq::util::PrimitiveMap::getShort (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Short value at the given *key*, if the *key* is not in the map or cannot be returned as the requested value then an exception of type *NoSuchElementException* is thrown.

Parameters

key - the location to return the value from.

Returns

the value at *key* in the type requested.

Exceptions

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.619.3.10 `virtual std::string activemq::util::PrimitiveMap::getString (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)` [virtual]

Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

key - the location to return the value from.

Returns

the value at key in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.619.3.11 `virtual void activemq::util::PrimitiveMap::setBool (const std::string & key, bool value)` [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.619.3.12 `virtual void activemq::util::PrimitiveMap::setByte (const std::string & key, unsigned char value)` [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.619.3.13 `virtual void activemq::util::PrimitiveMap::setByteArray (const std::string & key, const std::vector< unsigned char > & value)` [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.619.3.14 `virtual void activemq::util::PrimitiveMap::setChar (const std::string & key, char value) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.619.3.15 `virtual void activemq::util::PrimitiveMap::setDouble (const std::string & key, double value) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.619.3.16 `virtual void activemq::util::PrimitiveMap::setFloat (const std::string & key, float value) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.619.3.17 `virtual void activemq::util::PrimitiveMap::setInt (const std::string & key, int value) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.619.3.18 `virtual void activemq::util::PrimitiveMap::setLong (const std::string & key, long long value)` [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.619.3.19 `virtual void activemq::util::PrimitiveMap::setShort (const std::string & key, short value)` [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.619.3.20 `virtual void activemq::util::PrimitiveMap::setString (const std::string & key, const std::string & value)` [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.619.3.21 `std::string activemq::util::PrimitiveMap::toString ()` const

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

Returns

formatted String of all elements in the map.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveMap.h`

6.620 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

```
#include <src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h>
```

Public Member Functions

- **PrimitiveTypesMarshaller** ()
- virtual **~PrimitiveTypesMarshaller** ()

Static Public Member Functions

- static void **marshal** (const **util::PrimitiveMap** *map, std::vector< unsigned char > &buffer) throw (decaf::lang::Exception)
Marshal a primitive map object to the given byte buffer.
- static void **unmarshal** (**util::PrimitiveMap** *map, const std::vector< unsigned char > &buffer) throw (decaf::lang::Exception)
Unmarshal a PrimitiveMap from the provided Byte buffer.
- static void **marshal** (const **util::PrimitiveList** *list, std::vector< unsigned char > &buffer) throw (decaf::lang::Exception)
Marshal a primitive list object to the given byte buffer.
- static void **unmarshal** (**util::PrimitiveList** *list, const std::vector< unsigned char > &buffer) throw (decaf::lang::Exception)
Unmarshal a PrimitiveList from the provided byte buffer.
- static void **marshalMap** (const **util::PrimitiveMap** *map, **decaf::io::DataOutputStream** &dataOut) throw (decaf::lang::Exception)
Marshal a primitive map object to the given DataOutputStream.
- static **util::PrimitiveMap** * **unmarshalMap** (**decaf::io::DataInputStream** &dataIn) throw (decaf::lang::Exception)
Unmarshal a PrimitiveMap from the provided DataInputStream.
- static void **marshalList** (const **util::PrimitiveList** *list, **decaf::io::DataOutputStream** &dataOut) throw (decaf::lang::Exception)
Marshal a PrimitiveList to the given DataOutputStream.
- static **util::PrimitiveList** * **unmarshalList** (**decaf::io::DataInputStream** &dataIn) throw (decaf::lang::Exception)
Unmarshal a PrimitiveList from the given DataInputStream.

Static Protected Member Functions

- static void **marshalPrimitiveMap** (decaf::io::DataOutputStream &dataOut, const decaf::util::Map< std::string, util::PrimitiveValueNode > &map) throw (decaf::io::IOException)

Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.

- static void **marshalPrimitiveList** (decaf::io::DataOutputStream &dataOut, const decaf::util::List< util::PrimitiveValueNode > &list) throw (decaf::io::IOException)

Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.

- static void **marshalPrimitive** (decaf::io::DataOutputStream &dataOut, const util::PrimitiveValueNode &value) throw (decaf::io::IOException)

Used to Marshal the Primitive types out on the Wire.

- static void **unmarshalPrimitiveMap** (decaf::io::DataInputStream &dataIn, util::PrimitiveMap &map) throw (decaf::io::IOException)

Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.

- static void **unmarshalPrimitiveList** (decaf::io::DataInputStream &dataIn, decaf::util::StlList< util::PrimitiveValueNode > &list) throw (decaf::io::IOException)

Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.

- static util::PrimitiveValueNode **unmarshalPrimitive** (decaf::io::DataInputStream &dataIn) throw (decaf::io::IOException)

Unmarshals a Primitive Type from the stream, and returns it as a value Node.

6.620.1 Detailed Description

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

6.620.2 Constructor & Destructor Documentation

6.620.2.1 `activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::PrimitiveTypesMarshaller()` [inline]

6.620.2.2 `virtual activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::~~PrimitiveTypesMarshaller()` [inline, virtual]

6.620.3 Member Function Documentation

6.620.3.1 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal (const util::PrimitiveMap * map, std::vector< unsigned char > & buffer) throw (decaf::lang::Exception)` [static]

Marshal a primitive map object to the given byte buffer.

Parameters

map Map to Marshal.

buffer The byte buffer to write the marshaled data to.

Exceptions

Exception if an error occurs during the marshaling process.

6.620.3.2 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal (const util::PrimitiveList * list, std::vector< unsigned char > & buffer) throw (decaf::lang::Exception)` [static]

Marshal a primitive list object to the given byte buffer.

Parameters

map The PrimitiveList to Marshal.

buffer The byte buffer to write the marshaled data to.

Exceptions

Exception if an error occurs during the marshaling process.

6.620.3.3 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalList (const util::PrimitiveList * list, decaf::io::DataOutputStream & dataOut) throw (decaf::lang::Exception)` [static]

Marshal a PrimitiveList to the given DataOutputStream.

Parameters

list The list object to Marshal

dataOut Reference to a DataOutputStream to write the marshaled data to.

Exceptions

Exception if an error occurs during the marshaling process.

6.620.3.4 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalMap (const util::PrimitiveMap * *map*, decaf::io::DataOutputStream & *dataOut*) throw (decaf::lang::Exception) [static]

Marshal a primitive map object to the given DataOutputStream.

Parameters

map Map to Marshal.

dataOut Reference to a DataOutputStream to write the marshaled data to.

Exceptions

Exception if an error occurs during the marshaling process.

6.620.3.5 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitive (decaf::io::DataOutputStream & *dataOut*, const util::PrimitiveValueNode & *value*) throw (decaf::io::IOException) [static, protected]

Used to Marshal the Primitive types out on the Wire.

Parameters

dataOut - the DataOutputStream to write to

value - the ValueNode to write.

Exceptions

IOException

6.620.3.6 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveList (decaf::io::DataOutputStream & *dataOut*, const decaf::util::List< util::PrimitiveValueNode > & *list*) throw (decaf::io::IOException) [static, protected]

Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.

Parameters

dataOut - the DataOutputStream to write to

list - the ValueNode to write.

Exceptions

IOException

```
6.620.3.7 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveMap
( decaf::io::DataOutputStream & dataOut, const decaf::util::Map<
std::string, util::PrimitiveValueNode > & map ) throw (
decaf::io::IOException ) [static, protected]
```

Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.

Parameters

dataOut - the DataOutputStream to write to

map - the ValueNode to write.

Exceptions

IOException

```
6.620.3.8 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal
( util::PrimitiveList * list, const std::vector< unsigned char > & buffer
) throw ( decaf::lang::Exception ) [static]
```

Unmarshal a PrimitiveList from the provided byte buffer.

Parameters

map The List to populate with values from the marshaled data.

buffer The byte buffer containing the marshaled Map.

Exceptions

Exception if an error occurs during the unmarshal process.

```
6.620.3.9 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal
( util::PrimitiveMap * map, const std::vector< unsigned char > &
buffer ) throw ( decaf::lang::Exception ) [static]
```

Unmarshal a PrimitiveMap from the provided Byte buffer.

Parameters

map The Map to populate with values from the marshaled data.

buffer The byte buffer containing the marshaled Map.

Exceptions

Exception if an error occurs during the unmarshal process.

6.620.3.10 static util::PrimitiveList* activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalList (decaf::io::DataInputStream & *dataIn*) throw (decaf::lang::Exception) [static]

Unmarshal a PrimitiveList from the given DataInputStream.

Parameters

dataIn The DataInputStream instance to read the marshaled PrimitiveList from.

Returns

a pointer to a newly allocated PrimitiveList instnace.

Exceptions

Exception if an error occurs during the unmarshal process.

6.620.3.11 static util::PrimitiveMap* activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalMap (decaf::io::DataInputStream & *dataIn*) throw (decaf::lang::Exception) [static]

Unmarshal a PrimitiveMap from the provided DataInputStream.

Parameters

dataIn The DataInputStream instance to read the marshaled PrimitiveMap from.

Returns

a pointer to a newly allocated PrimitiveMap instnace.

Exceptions

Exception if an error occurs during the unmarshal process.

6.620.3.12 static util::PrimitiveValueNode activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitive (decaf::io::DataInputStream & *dataIn*) throw (decaf::io::IOException) [static, protected]

Unmarshals a Primitive Type from the stream, and returns it as a value Node.

Parameters

dataIn - DataInputStream to read from.

Returns

a PrimitiveValueNode containing the data.

Exceptions

IOException

6.620.3.13 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveList (decaf::io::DataInputStream & *dataIn*, decaf::util::StlList< util::PrimitiveValueNode > & *list*) throw (decaf::io::IOException) [static, protected]

Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.

Parameters

dataIn - DataInputStream to read from.

list - the ValueNode to write.

Exceptions

IOException

6.620.3.14 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveMap (decaf::io::DataInputStream & *dataIn*, util::PrimitiveMap & *map*) throw (decaf::io::IOException) [static, protected]

Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.

Parameters

dataIn - DataInputStream to read from.

map - the map to fill with data.

Exceptions

IOException

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h

6.621 activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference

Define a union type comprised of the various types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```


Data Fields

- bool **boolValue**
- unsigned char **byteValue**
- char **charValue**
- short **shortValue**
- int **intValue**
- long long **longValue**
- double **doubleValue**
- float **floatValue**
- std::string * **stringValue**
- std::vector< unsigned char > * **byteArrayValue**
- decaf::util::List< PrimitiveValueNode > * **listValue**
- decaf::util::Map< std::string, PrimitiveValueNode > * **mapValue**

6.621.1 Detailed Description

Define a union type comprised of the various types.

6.621.2 Field Documentation

- 6.621.2.1** bool **activemq::util::PrimitiveValueNode::PrimitiveValue::boolValue**
- 6.621.2.2** std::vector<unsigned char>* **activemq::util::PrimitiveValueNode::PrimitiveValue::byteArrayValue**
- 6.621.2.3** unsigned char **activemq::util::PrimitiveValueNode::PrimitiveValue::byteValue**
- 6.621.2.4** char **activemq::util::PrimitiveValueNode::PrimitiveValue::charValue**
- 6.621.2.5** double **activemq::util::PrimitiveValueNode::PrimitiveValue::doubleValue**
- 6.621.2.6** float **activemq::util::PrimitiveValueNode::PrimitiveValue::floatValue**
- 6.621.2.7** int **activemq::util::PrimitiveValueNode::PrimitiveValue::intValue**
- 6.621.2.8** decaf::util::List<PrimitiveValueNode>* **activemq::util::PrimitiveValueNode::PrimitiveValue::listValue**
- 6.621.2.9** long long **activemq::util::PrimitiveValueNode::PrimitiveValue::longValue**
- 6.621.2.10** decaf::util::Map<std::string, PrimitiveValueNode>* **activemq::util::PrimitiveValueNode::PrimitiveValue::mapValue**
- 6.621.2.11** short **activemq::util::PrimitiveValueNode::PrimitiveValue::shortValue**
- 6.621.2.12** std::string* **activemq::util::PrimitiveValueNode::PrimitiveValue::stringValue**

The documentation for this union was generated from the following file:

- `src/main/activemq/util/PrimitiveValueNode.h`

6.622 `activemq::util::PrimitiveValueConverter` Class Reference

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2817) from one type to another.

```
#include <src/main/activemq/util/PrimitiveValueConverter.h>
```

Public Member Functions

- **PrimitiveValueConverter** ()
- virtual **~PrimitiveValueConverter** ()
- `template<typename TO >`
`TO convert (const PrimitiveValueNode &value) const throw (decaf::lang::exceptions::UnsupportedOperationException)`

6.622.1 Detailed Description

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2817) from one type to another. If the conversion is supported then calling the `convert` method will throw an `UnsupportedOperationException` to indicate that its not possible to perform the conversion.

This class is used to implement the rules of conversion on CMS Message properties, the following conversion table must be implemented. A value written as the row type can be read in the column type.

		boolean	byte	short	int	long	float	double	String	-----								
boolean		X X	byte		X X X X X	short		X X X X	int		X X X	long		X X	float		X X X	double
	X X	String		X X X X X X X X		-----												

Since

3.0

6.622.2 Constructor & Destructor Documentation

6.622.2.1 `activemq::util::PrimitiveValueConverter::PrimitiveValueConverter ()`
[inline]

6.622.2.2 `virtual`
`activemq::util::PrimitiveValueConverter::~~PrimitiveValueConverter ()`
[inline, virtual]

6.622.3 Member Function Documentation

6.622.3.1 `std::vector< unsigned char > ac-`
`tivemq::util::PrimitiveValueConverter::convert (const`
`PrimitiveValueNode & value) const throw (`
`decaf::lang::exceptions::UnsupportedOperationException)` [inline]

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveValueConverter.h`

6.623 activemq::util::PrimitiveValueNode Class Reference

Class that wraps around a single value of one of the many types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

Data Structures

- union **PrimitiveValue**
Define a union type comprised of the various types.

Public Types

- enum **PrimitiveType** {
 NULL_TYPE = 0, **BOOLEAN_TYPE** = 1, **BYTE_TYPE** = 2, **CHAR_TYPE** = 3,
 SHORT_TYPE = 4, **INTEGER_TYPE** = 5, **LONG_TYPE** = 6, **DOUBLE_TYPE** = 7,
 FLOAT_TYPE = 8, **STRING_TYPE** = 9, **BYTE_ARRAY_TYPE** = 10,
 MAP_TYPE = 11,
 LIST_TYPE = 12, **BIG_STRING_TYPE** = 13 }
Enumeration for the various primitive types.

Public Member Functions

- **PrimitiveValueNode** ()
Default Constructor, creates a value of the NULL_TYPE.

- **PrimitiveValueNode** (bool value)
Boolean Value Constructor.
- **PrimitiveValueNode** (unsigned char value)
Byte Value Constructor.
- **PrimitiveValueNode** (char value)
Char Value Constructor.
- **PrimitiveValueNode** (short value)
Short Value Constructor.
- **PrimitiveValueNode** (int value)
Int Value Constructor.
- **PrimitiveValueNode** (long long value)
Long Value Constructor.
- **PrimitiveValueNode** (float value)
Float Value Constructor.
- **PrimitiveValueNode** (double value)
Double Value Constructor.
- **PrimitiveValueNode** (const char *value)
String Value Constructor.
- **PrimitiveValueNode** (const std::string &value)
String Value Constructor.
- **PrimitiveValueNode** (const std::vector< unsigned char > &value)
Byte Array Value Constructor.
- **PrimitiveValueNode** (const decaf::util::List< PrimitiveValueNode > &value)
Primitive List Constructor.
- **PrimitiveValueNode** (const decaf::util::Map< std::string, PrimitiveValueNode > &value)
Primitive Map Value Constructor.
- **PrimitiveValueNode** (const PrimitiveValueNode &node)
Copy constructor.
- **~PrimitiveValueNode** ()
- **PrimitiveValueNode & operator=** (const PrimitiveValueNode &node)
Assignment operator, copies the data from the other node.
- bool **operator==** (const PrimitiveValueNode &node) const
Comparison Operator, compares this node to the other node.

- **PrimitiveType** **getType** () const
Gets the Value Type of this type wrapper.
- **PrimitiveValue** **getValue** () const
Gets the internal Primitive Value object from this wrapper.
- void **setValue** (const **PrimitiveValue** &value, **PrimitiveType** valueType)
Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.
- void **clear** ()
Clears the value from this wrapper converting it back to a blank NULL_ TYPE value.
- void **setBool** (bool value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- bool **getBool** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Boolean value of this Node.
- void **setByte** (unsigned char value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- unsigned char **getByte** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Byte value of this Node.
- void **setChar** (char value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- char **getChar** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Character value of this Node.
- void **setShort** (short value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- short **getShort** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Short value of this Node.
- void **setInt** (int value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- int **getInt** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Integer value of this Node.
- void **setLong** (long long value)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- long long **getLong** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Long value of this Node.
- void **setFloat** (float value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- float **getFloat** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Float value of this Node.
- void **setDouble** (double value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- double **getDouble** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Double value of this Node.
- void **setString** (const std::string &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- std::string **getString** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the String value of this Node.
- void **setByteArray** (const std::vector< unsigned char > &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- std::vector< unsigned char > **getByteArray** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Byte Array value of this Node.
- void **setList** (const decaf::util::List< PrimitiveValueNode > &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- const decaf::util::List< PrimitiveValueNode > & **getList** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Primitive List value of this Node.
- void **setMap** (const decaf::util::Map< std::string, PrimitiveValueNode > &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- const decaf::util::Map< std::string, PrimitiveValueNode > & **getMap** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Primitive Map value of this Node.

- `std::string toString () const`
Creates a string representation of this value.

6.623.1 Detailed Description

Class that wraps around a single value of one of the many types. Manages memory for complex types, such as strings. Note: the destructor was left non-virtual so no virtual table will be created. This probably isn't necessary, but will avoid needless memory allocation. Since we'll never extend this class, not having a virtual destructor isn't a concern.

6.623.2 Member Enumeration Documentation

6.623.2.1 `enum activemq::util::PrimitiveValueNode::PrimitiveType`

Enumeration for the various primitive types.

Enumerator:

NULL_TYPE
BOOLEAN_TYPE
BYTE_TYPE
CHAR_TYPE
SHORT_TYPE
INTEGER_TYPE
LONG_TYPE
DOUBLE_TYPE
FLOAT_TYPE
STRING_TYPE
BYTE_ARRAY_TYPE
MAP_TYPE
LIST_TYPE
BIG_STRING_TYPE

6.623.3 Constructor & Destructor Documentation

6.623.3.1 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ()`

Default Constructor, creates a value of the `NULL_TYPE`.

6.623.3.2 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (bool value)`

Boolean Value Constructor.

Parameters

value - the new value to store.

6.623.3.3 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (unsigned char value)`

Byte Value Constructor.

Parameters

value - the new value to store.

6.623.3.4 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (char value)`

Char Value Constructor.

Parameters

value - the new value to store.

6.623.3.5 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (short value)`

Short Value Constructor.

Parameters

value - the new value to store.

6.623.3.6 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (int value)`

Int Value Constructor.

Parameters

value - the new value to store.

6.623.3.7 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (long long value)`

Long Value Constructor.

Parameters

value - the new value to store.

6.623.3.8 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (float value)`

Float Value Constructor.

Parameters

value - the new value to store.

6.623.3.9 activemq::util::PrimitiveValueNode::PrimitiveValueNode (double *value*)

Double Value Constructor.

Parameters

value - the new value to store.

6.623.3.10 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const char * *value*)

String Value Constructor.

Parameters

value - the new value to store.

6.623.3.11 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::string & *value*)

String Value Constructor.

Parameters

value - the new value to store.

6.623.3.12 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::vector< unsigned char > & *value*)

Byte Array Value Constructor.

Parameters

value - the new value to store.

6.623.3.13 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::List< PrimitiveValueNode > & *value*)

Primitive List Constructor.

Parameters

value - the new value to store.

6.623.3.14 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::Map< std::string, PrimitiveValueNode > & *value*)

Primitive Map Value Constructor.

Parameters

value - the new value to store.

6.623.3.15 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const PrimitiveValueNode & node)`

Copy constructor.

Parameters

node The instance of another node to copy to this one.

6.623.3.16 `activemq::util::PrimitiveValueNode::~~PrimitiveValueNode ()`
[inline]

6.623.4 Member Function Documentation

6.623.4.1 `void activemq::util::PrimitiveValueNode::clear ()`

Clears the value from this wrapper converting it back to a blank NULL_TYPE value.

6.623.4.2 `bool activemq::util::PrimitiveValueNode::getBool () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Boolean value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.623.4.3 `unsigned char activemq::util::PrimitiveValueNode::getBytes () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Byte value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.623.4.4 `std::vector<unsigned char> activemq::util::PrimitiveValueNode::getBytesArray () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Byte Array value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.623.4.5 `char activemq::util::PrimitiveValueNode::getChar () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Character value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.623.4.6 `double activemq::util::PrimitiveValueNode::getDouble () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Double value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.623.4.7 `float activemq::util::PrimitiveValueNode::getFloat () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Float value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.623.4.8 `int activemq::util::PrimitiveValueNode::getInt () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Integer value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

```
6.623.4.9  const decaf::util::List<PrimitiveValueNode>&
            activemq::util::PrimitiveValueNode::getList (    ) const throw (
            decaf::lang::exceptions::NoSuchElementException )
```

Gets the Primitive List value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

```
6.623.4.10 long long activemq::util::PrimitiveValueNode::getLong (    ) const throw
            ( decaf::lang::exceptions::NoSuchElementException )
```

Gets the Long value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

```
6.623.4.11 const decaf::util::Map<std::string, PrimitiveValueNode>&
            activemq::util::PrimitiveValueNode::getMap (    ) const throw (
            decaf::lang::exceptions::NoSuchElementException )
```

Gets the Primitive Map value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.623.4.12 `short activemq::util::PrimitiveValueNode::getShort () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Short value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.623.4.13 `std::string activemq::util::PrimitiveValueNode::getString () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the String value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.623.4.14 `PrimitiveType activemq::util::PrimitiveValueNode::getType () const [inline]`

Gets the Value Type of this type wrapper.

Returns

the PrimitiveType value for this wrapper.

6.623.4.15 `PrimitiveValue activemq::util::PrimitiveValueNode::getValue () const [inline]`

Gets the internal Primitive Value object from this wrapper.

Returns

a copy of the contained **PrimitiveValue** (p. 2814)

6.623.4.16 `PrimitiveValueNode& activemq::util::PrimitiveValueNode::operator= (const PrimitiveValueNode & node)`

Assignment operator, copies the data from the other node.

Parameters

node The instance of another node to copy to this one.

6.623.4.17 `bool activemq::util::PrimitiveValueNode::operator==(const PrimitiveValueNode & node) const`

Comparison Operator, compares this node to the other node.

Returns

true if the values are the same false otherwise.

6.623.4.18 `void activemq::util::PrimitiveValueNode::setBool (bool value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.19 `void activemq::util::PrimitiveValueNode::setByte (unsigned char value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.20 `void activemq::util::PrimitiveValueNode::setByteArray (const std::vector< unsigned char > & value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.21 `void activemq::util::PrimitiveValueNode::setChar (char value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.22 void activemq::util::PrimitiveValueNode::setDouble (double *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.23 void activemq::util::PrimitiveValueNode::setFloat (float *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.24 void activemq::util::PrimitiveValueNode::setInt (int *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.25 void activemq::util::PrimitiveValueNode::setList (const decaf::util::List< PrimitiveValueNode > & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.26 void activemq::util::PrimitiveValueNode::setLong (long long *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.27 `void activemq::util::PrimitiveValueNode::setMap (const
decaf::util::Map< std::string, PrimitiveValueNode > & value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.28 `void activemq::util::PrimitiveValueNode::setShort (short value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.29 `void activemq::util::PrimitiveValueNode::setString (const std::string &
value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.623.4.30 `void activemq::util::PrimitiveValueNode::setValue (const
PrimitiveValue & value, PrimitiveType valueType)`

Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.

Parameters

value The value to set as the value contained in this Node.

valueType The type of the value being set into this one.

6.623.4.31 `std::string activemq::util::PrimitiveValueNode::toString () const`

Creates a string representation of this value.

Returns

string value of this type wrapper.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveValueNode.h`

6.624 decaf::security::Principal Class Reference

Base interface for a principal, which can represent an individual or organization.

```
#include <src/main/decaf/security/Principal.h>
```

Inheritance diagram for decaf::security::Principal:

Public Member Functions

- virtual **~Principal** ()
- virtual bool **equals** (const **Principal** &another) const =0
Compares two principals to see if they are the same.
- virtual std::string **getName** () const =0
Provides the name of this principal.

6.624.1 Detailed Description

Base interface for a principal, which can represent an individual or organization.

6.624.2 Constructor & Destructor Documentation

6.624.2.1 virtual decaf::security::Principal::~~Principal () [inline, virtual]

6.624.3 Member Function Documentation

6.624.3.1 virtual bool decaf::security::Principal::equals (const **Principal** &
another) const [pure virtual]

Compares two principals to see if they are the same.

Parameters

another A principal to be tested for equality to this one.

Returns

true if the given principal is equivalent to this one.

6.624.3.2 virtual std::string decaf::security::Principal::getName () const [pure virtual]

Provides the name of this principal.

Returns

the name of this principal.

Implemented in **decaf::security::auth::x500::X500Principal** (p. 3762).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/Principal.h`

6.625 decaf::util::PriorityQueue< E > Class Template Reference

An unbounded priority queue based on a binary heap algorithm.

```
#include <src/main/decaf/util/PriorityQueue.h>
```

Inheritance diagram for `decaf::util::PriorityQueue< E >`:

Data Structures

- class **ConstPriorityQueueIterator**
- class **PriorityQueueIterator**

Public Member Functions

- **PriorityQueue** ()
*Creates a **Priority Queue** (p. 2948) with the default initial capacity.*
- **PriorityQueue** (std::size_t initialCapacity)
*Creates a **Priority Queue** (p. 2948) with the capacity value supplied.*
- **PriorityQueue** (std::size_t initialCapacity, **Comparator**< E > *comparator)
*Creates a **Priority Queue** (p. 2948) with the default initial capacity.*
- **PriorityQueue** (const **Collection**< E > &source)
*Creates a **PriorityQueue** (p. 2832) containing the elements in the specified **Collection** (p. 1097).*
- **PriorityQueue** (const **PriorityQueue**< E > &source)
*Creates a **PriorityQueue** (p. 2832) containing the elements in the specified priority queue.*
- virtual **~PriorityQueue** ()
- **PriorityQueue**< E > & **operator=** (const **Collection**< E > &source)
*Assignment operator, assign another **Collection** (p. 1097) to this one.*
- **PriorityQueue**< E > & **operator=** (const **PriorityQueue**< E > &source)
*Assignment operator, assign another **PriorityQueue** (p. 2832) to this one.*
- virtual **decaf::util::Iterator**< E > * **iterator** ()
- virtual **decaf::util::Iterator**< E > * **iterator** () const
- virtual std::size_t **size** () const

Returns the number of elements in this collection.

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
Removes all elements of the queue.
- virtual bool **offer** (const E &value) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Inserts the specified element into the queue provided that the condition allows such an operation.
- virtual bool **poll** (E &result)
Gets and removes the element in the head of the queue.
- virtual bool **peek** (E &result) const
Gets but not removes the element in the head of the queue.
- virtual E **remove** () throw (decaf::lang::exceptions::NoSuchElementException)
Retrieves and removes the head of this queue.
- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes a single instance of the specified element from this collection, if it is present (optional operation).
- virtual bool **add** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)
Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.
- decaf::lang::Pointer< Comparator< E > > **comparator** () const
*obtains a Copy of the Pointer instance that this **PriorityQueue** (p. 2832) is using to compare the elements in the queue with.*

Friends

- class **PriorityQueueIterator**

6.625.1 Detailed Description

template<typename E> class decaf::util::PriorityQueue< E >

An unbounded priority queue based on a binary heap algorithm. The elements of the priority queue are ordered according to their natural ordering, or by a **Comparator** (p. 1127) provided to one of the constructors that accepts Comparators. A priority queue relying on natural ordering also does not permit insertion of non-comparable objects (doing so may result in a compiler error).

The head of this queue is the least element with respect to the specified ordering. If multiple elements are tied for least value, the head is one of those elements -- ties are broken arbitrarily. The queue retrieval operations poll, remove, peek, and element access the element at the head of the queue.

A priority queue is unbounded, but has an internal capacity governing the size of an array used to store the elements on the queue. It is always at least as large as the queue size. As elements are added to a priority queue, its capacity grows automatically. The details of the growth policy are not specified.

This class and its iterator implement all of the optional methods of the **Collection** (p. 1097) and **Iterator** (p. 2012) interfaces. The **Iterator** (p. 2012) provided in method **iterator()** (p. 2836) is not guaranteed to traverse the elements of the priority queue in any particular order. If you need ordered traversal, consider using `Arrays::sort(pq.toArray())`.

Note that this implementation is not synchronized. Multiple threads should not access a **PriorityQueue** (p. 2832) instance concurrently if any of the threads modifies the queue. Instead, use the thread-safe `PriorityBlockingQueue` class.

Implementation note: this implementation provides $O(\log(n))$ time for the enqueueing and dequeuing methods (`offer`, `poll`, **`remove()`** (p. 2838) and `add`); linear time for the `remove(Object)` and `contains(Object)` methods; and constant time for the retrieval methods (`peek`, `element`, and `size`).

Since

1.0

6.625.2 Constructor & Destructor Documentation

6.625.2.1 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue () [inline]`

Creates a **Priority Queue** (p. 2948) with the default initial capacity.

6.625.2.2 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (std::size_t initialCapacity) [inline]`

Creates a **Priority Queue** (p. 2948) with the capacity value supplied.

Parameters

initialCapacity The initial number of elements allocated to this **PriorityQueue** (p. 2832).

6.625.2.3 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (std::size_t initialCapacity, Comparator< E > * comparator) [inline]`

Creates a **Priority Queue** (p. 2948) with the default initial capacity.

This new **PriorityQueue** (p. 2832) takes ownership of the passed **Comparator** (p. 1127) instance and uses that to determine the ordering of the elements in the **Queue** (p. 2948).

Parameters

initialCapacity The initial number of elements allocated to this **PriorityQueue** (p. 2832).

comparator The **Comparator** (p. 1127) instance to use in sorting the elements in the **Queue** (p. 2948).

Exceptions

NullPointerException if the passed **Comparator** (p. 1127) is NULL.

6.625.2.4 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (const Collection< E > & source) [inline]`

Creates a **PriorityQueue** (p. 2832) containing the elements in the specified **Collection** (p. 1097).

Parameters

source the **Collection** (p. 1097) whose elements are to be placed into this priority queue

6.625.2.5 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (const PriorityQueue< E > & source) [inline]`

Creates a **PriorityQueue** (p. 2832) containing the elements in the specified priority queue.

This priority queue will be ordered according to the same ordering as the given priority queue.

Parameters

source the priority queue whose elements are to be placed into this priority queue

6.625.2.6 `template<typename E> virtual decaf::util::PriorityQueue< E >::~PriorityQueue () [inline, virtual]`

6.625.3 Member Function Documentation

6.625.3.1 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::add (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]`

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an **IllegalStateException** if no space is currently available.

This implementation returns true if offer succeeds, else throws an **IllegalStateException**.

Parameters

value - the element to offer to the **Queue** (p. 2948).

Returns

true if the add succeeds.

Exceptions

IllegalArgumentException if the element cannot be added.

Reimplemented from **decaf::util::AbstractQueue**< **E** > (p. 159).

References **DECAF_CATCH_EXCEPTION_CONVERT**, **DECAF_CATCH_RETHROW**, **DECAF_CATCHALL_THROW**, and **decaf::util::PriorityQueue**< **E** >::offer().

6.625.3.2 `template<typename E> virtual void decaf::util::PriorityQueue< E >::clear () throw (lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes all elements of the queue.

This implementation repeatedly invokes poll until it returns the empty marker.

Reimplemented from **decaf::util::AbstractQueue**< **E** > (p. 160).

6.625.3.3 `template<typename E> decaf::lang::Pointer< Comparator<E> > decaf::util::PriorityQueue< E >::comparator () const [inline]`

obtains a Copy of the Pointer instance that this **PriorityQueue** (p. 2832) is using to compare the elements in the queue with.

The returned value is a copy, the caller cannot change the value if the internal Pointer value.

Returns

a copy of the **Comparator** (p. 1127) Pointer being used by this **Queue** (p. 2948).

6.625.3.4 `template<typename E> virtual decaf::util::Iterator<E>* decaf::util::PriorityQueue< E >::iterator () const [inline, virtual]`

6.625.3.5 `template<typename E> virtual decaf::util::Iterator<E>* decaf::util::PriorityQueue< E >::iterator () [inline, virtual]`

References **decaf::util::PriorityQueue**< **E** >::PriorityQueueIterator.

6.625.3.6 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::offer (const E & value) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException) [inline, virtual]`

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the **collection.add(E)**, since the latter might throw an exception if the operation fails.

Parameters

value the specified element to insert into the queue.

Returns

true if the operation succeeds and false if it fails.

Exceptions

NullPointerException if the **Queue** (p. 2948) implementation does not allow Null values to be inserted into the **Queue** (p. 2948).

IllegalArgumentException if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 2949).

Referenced by **decaf::util::PriorityQueue< E >::add()**.

6.625.3.7 `template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue< E >::operator= (const PriorityQueue< E > & source) [inline]`

Assignment operator, assign another **PriorityQueue** (p. 2832) to this one.

Parameters

source The **PriorityQueue** (p. 2832) to copy to this one.

6.625.3.8 `template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue< E >::operator= (const Collection< E > & source) [inline]`

Assignment operator, assign another **Collection** (p. 1097) to this one.

Parameters

source The **Collection** (p. 1097) to copy to this one.

6.625.3.9 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::peek (E & result) const [inline, virtual]`

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

Parameters

result Reference to an instance of the contained type to assigned the removed value to.

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2950).

6.625.3.10 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::poll (E & result) [inline, virtual]`

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 2948) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

Parameters

result Reference to an instance of the contained type to assigned the removed value to.

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2950).

6.625.3.11 `template<typename E> virtual E decaf::util::PriorityQueue< E
>::remove () throw (decaf::lang::exceptions::NoSuchElementException
) [inline, virtual]`

Retrieves and removes the head of this queue.

This method differs from poll only in that it throws an exception if this queue is empty.

This implementation returns the result of poll unless the queue is empty.

Returns

a copy of the element in the head of the queue.

Exceptions

NoSuchElementException if the queue is empty.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 161).

6.625.3.12 `template<typename E> virtual bool decaf::util::PriorityQueue<
E >::remove (const E & value) throw (
lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 1097).

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 151).

6.625.3.13 `template<typename E> virtual std::size_t decaf::util::PriorityQueue<E>::size () const [inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX_VALUE elements, returns Integer.MAX_VALUE.

Returns

the number of elements in this collection

Implements **decaf::util::Collection**< **E** > (p. 1106).

6.625.4 Friends And Related Function Documentation

6.625.4.1 `template<typename E> friend class PriorityQueueIterator [friend]`

Referenced by `decaf::util::PriorityQueue< E >::iterator()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/PriorityQueue.h`

6.626 activemq::commands::ProducerAck Class Reference

```
#include <src/main/activemq/commands/ProducerAck.h>
```

Inheritance diagram for `activemq::commands::ProducerAck`:

Public Member Functions

- **ProducerAck** ()
- virtual **~ProducerAck** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ProducerAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual int **getSize** () const
- virtual void **setSize** (int size)
- virtual bool **isProducerAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_PRODUCERACK** = 19

Protected Attributes

- **Pointer**< **ProducerId** > producerId
- int size

6.626.1 Constructor & Destructor Documentation

6.626.1.1 activemq::commands::ProducerAck::ProducerAck ()

6.626.1.2 virtual activemq::commands::ProducerAck::~~ProducerAck ()
[virtual]

6.626.2 Member Function Documentation

6.626.2.1 virtual **ProducerAck*** **activemq::commands::ProducerAck::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.626.2.2 `virtual void activemq::commands::ProducerAck::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.626.2.3 `virtual bool activemq::commands::ProducerAck::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.626.2.4 `virtual unsigned char activemq::commands::ProducerAck::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1553) type copy.

Implements `activemq::commands::DataStructure` (p. 1557).

6.626.2.5 `virtual const Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId () const [virtual]`

6.626.2.6 `virtual Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId () [virtual]`

6.626.2.7 `virtual int activemq::commands::ProducerAck::getSize () const [virtual]`

6.626.2.8 `virtual bool activemq::commands::ProducerAck::isProducerAck () const [inline, virtual]`

Returns

an answer of true to the `isProducerAck()` (p. 2841) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 699).

6.626.2.9 `virtual void activemq::commands::ProducerAck::setProducerId (const Pointer< ProducerId > & producerId) [virtual]`

6.626.2.10 `virtual void activemq::commands::ProducerAck::setSize (int size) [virtual]`

6.626.2.11 `virtual std::string activemq::commands::ProducerAck::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p.700).

6.626.2.12 `virtual Pointer<Command> activemq::commands::ProducerAck::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p.3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1112).

6.626.3 Field Documentation

6.626.3.1 `const unsigned char activemq::commands::ProducerAck::ID_-PRODUCERACK = 19 [static]`

6.626.3.2 `Pointer<ProducerId> activemq::commands::ProducerAck::producerId [protected]`

6.626.3.3 `int activemq::commands::ProducerAck::size [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerAck.h`

6.627 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p.2842).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.627.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p.2842). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.627.2 Constructor & Destructor Documentation

6.627.2.1 `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::ProducerAckMarshaller () [inline]`

6.627.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::~~ProducerAckMarshaller () [inline, virtual]`

6.627.3 Member Function Documentation

6.627.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.627.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.627.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.627.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 737).

6.627.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 738).

6.627.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 739).

```
6.627.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h`

6.628 **activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2846).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller**:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.628.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p.2846). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.628.2 Constructor & Destructor Documentation

6.628.2.1 `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::ProducerAckMarshaller () [inline]`

6.628.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::~~ProducerAckMarshaller () [inline, virtual]`

6.628.3 Member Function Documentation

6.628.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.628.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.628.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.628.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 710).

6.628.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 711).

6.628.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.628.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h`

6.629 `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` Class Reference

Marshaling code for Open Wire Format for `ProducerAckMarshaller` (p. 2850).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller`:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.629.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p.2850). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.629.2 Constructor & Destructor Documentation

6.629.2.1 `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::ProducerAckMarshaller () [inline]`

6.629.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::~~ProducerAckMarshaller () [inline, virtual]`

6.629.3 Member Function Documentation

6.629.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.629.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.629.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.629.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 703).

6.629.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 704).

6.629.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.629.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h`

6.630 `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` Class Reference

Marshaling code for Open Wire Format for `ProducerAckMarshaller` (p. 2854).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller`:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.630.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p.2854). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.630.2 Constructor & Destructor Documentation

6.630.2.1 `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::ProducerAckMarshaller () [inline]`

6.630.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::~~ProducerAckMarshaller () [inline, virtual]`

6.630.3 Member Function Documentation

6.630.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.630.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.630.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.630.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 723).

6.630.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 724).

6.630.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.630.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h`

6.631 `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` Class Reference

Marshaling code for Open Wire Format for `ProducerAckMarshaller` (p. 2858).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller`:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.631.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p.2858). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.631.2 Constructor & Destructor Documentation

6.631.2.1 `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::ProducerAckMarshaller () [inline]`

6.631.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::~~ProducerAckMarshaller () [inline, virtual]`

6.631.3 Member Function Documentation

6.631.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.631.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.631.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.631.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 730).

6.631.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 731).

6.631.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

```
6.631.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h`

6.632 `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` Class Reference

Marshaling code for Open Wire Format for `ProducerAckMarshaller` (p. 2862).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller`:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.632.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p.2862). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.632.2 Constructor & Destructor Documentation

6.632.2.1 `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::ProducerAckMarshaller () [inline]`

6.632.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::~~ProducerAckMarshaller () [inline, virtual]`

6.632.3 Member Function Documentation

6.632.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.632.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.632.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.632.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 717).

6.632.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 718).

6.632.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

```
6.632.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h`

6.633 `activemq::cmsutil::ProducerCallback` Class Reference

Callback for sending a message to a CMS destination.

```
#include <src/main/activemq/cmsutil/ProducerCallback.h>
```

Inheritance diagram for `activemq::cmsutil::ProducerCallback`:

Public Member Functions

- virtual `~ProducerCallback` ()
- virtual void `doInCms` (`cms::Session` *session, `cms::MessageProducer` *producer)=0
throw (`cms::CMSEException`)

Execute an action given a session and producer.

6.633.1 Detailed Description

Callback for sending a message to a CMS destination.

6.633.2 Constructor & Destructor Documentation

- 6.633.2.1** virtual `activemq::cmsutil::ProducerCallback::~ProducerCallback` ()
[inline, virtual]

6.633.3 Member Function Documentation

- 6.633.3.1** virtual void `activemq::cmsutil::ProducerCallback::doInCms` (
`cms::Session` * *session*, `cms::MessageProducer` * *producer*) throw (
`cms::CMSEException`) [pure virtual]

Execute an action given a session and producer.

Parameters

session the CMS Session

producer the CMS Producer

Exceptions

cms::CMSEException (p. 1074) if thrown by CMS API methods

Implemented in `activemq::cmsutil::CmsTemplate::SendExecutor` (p. 3136).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/ProducerCallback.h`

6.634 activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate::ProducerExecutor`:

Public Member Functions

- **ProducerExecutor** (**ProducerCallback** **action*, **CmsTemplate** **parent*, **cms::Destination** **destination*)
- virtual **~ProducerExecutor** ()
- virtual void **doInCms** (**cms::Session** **session*) throw (**cms::CMSException**)
Execute any number of operations against the supplied CMS session.
- virtual **cms::Destination** * **getDestination** (**cms::Session** **session* **AMQCPP_** - **UNUSED**) throw (**cms::CMSException**)

Protected Member Functions

- **ProducerExecutor** (const **ProducerExecutor** &)
- **ProducerExecutor** & **operator=** (const **ProducerExecutor** &)

Protected Attributes

- **ProducerCallback** * *action*
- **CmsTemplate** * *parent*
- **cms::Destination** * *destination*

6.634.1 Constructor & Destructor Documentation

- 6.634.1.1** **activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor** (const **ProducerExecutor** &) [inline, protected]
- 6.634.1.2** **activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor** (**ProducerCallback** * *action*, **CmsTemplate** * *parent*, **cms::Destination** * *destination*) [inline]
- 6.634.1.3** virtual **activemq::cmsutil::CmsTemplate::ProducerExecutor::~~ProducerExecutor** () [inline, virtual]

6.634.2 Member Function Documentation

- 6.634.2.1** virtual void **activemq::cmsutil::CmsTemplate::ProducerExecutor::doInCms** (**cms::Session** * *session*) throw (**cms::CMSException**) [virtual]

Execute any number of operations against the supplied CMS session.

Parameters

session the CMS Session

Exceptions

cms::CMSException (p. 1074) if thrown by CMS API methods

Implements **activemq::cmsutil::SessionCallback** (p. 3161).

- 6.634.2.2** virtual cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::getDestination (cms::Session *session *AMQCPP_UNUSED*) throw (cms::CMSException) [inline, virtual]
- 6.634.2.3** ProducerExecutor& activemq::cmsutil::CmsTemplate::ProducerExecutor::operator= (const ProducerExecutor &) [inline, protected]
- 6.634.3** Field Documentation
- 6.634.3.1** ProducerCallback* activemq::cmsutil::CmsTemplate::ProducerExecutor::action [protected]
- 6.634.3.2** cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::destination [protected]
- 6.634.3.3** CmsTemplate* activemq::cmsutil::CmsTemplate::ProducerExecutor::parent [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/CmsTemplate.h

6.635 activemq::commands::ProducerId Class Reference

```
#include <src/main/activemq/commands/ProducerId.h>
```

Inheritance diagram for activemq::commands::ProducerId:

Public Types

- typedef decaf::lang::PointerComparator< ProducerId > COMPARATOR

Public Member Functions

- **ProducerId** ()
- **ProducerId** (const **ProducerId** &other)
- **ProducerId** (const **SessionId** &sessionId, long long consumerId)
- **ProducerId** (std::string producerId)
- virtual ~**ProducerId** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ProducerId** * **cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

- const **Pointer**< **SessionId** > & **getParentId** () const
- void **setProducerSessionKey** (std::string sessionKey)
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual int **compareTo** (const **ProducerId** &value) const
- virtual bool **equals** (const **ProducerId** &value) const
- virtual bool **operator==** (const **ProducerId** &value) const
- virtual bool **operator<** (const **ProducerId** &value) const
- **ProducerId** & **operator=** (const **ProducerId** &other)

Static Public Attributes

- static const unsigned char **ID_PRODUCERID** = 123

Protected Attributes

- std::string **connectionId**
- long long **value**
- long long **sessionId**

6.635.1 Member Typedef Documentation

6.635.1.1 `typedef decaf::lang::PointerComparator<ProducerId>
activemq::commands::ProducerId::COMPARATOR`

6.635.2 Constructor & Destructor Documentation

6.635.2.1 `activemq::commands::ProducerId::ProducerId ()`

6.635.2.2 `activemq::commands::ProducerId::ProducerId (const ProducerId &
other)`

6.635.2.3 `activemq::commands::ProducerId::ProducerId (const SessionId &
sessionId, long long consumerId)`

6.635.2.4 `activemq::commands::ProducerId::ProducerId (std::string producerId)`

6.635.2.5 `virtual activemq::commands::ProducerId::~~ProducerId () [virtual]`

6.635.3 Member Function Documentation

6.635.3.1 `virtual ProducerId* ac-
tivemq::commands::ProducerId::cloneDataStructure ()
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.635.3.2 `virtual int activemq::commands::ProducerId::compareTo (const
ProducerId & value) const [virtual]`

6.635.3.3 `virtual void activemq::commands::ProducerId::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.635.3.4 `virtual bool activemq::commands::ProducerId::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.635.3.5 virtual bool activemq::commands::ProducerId::equals (const ProducerId
& *value*) const [virtual]

6.635.3.6 virtual const std::string& ac-
tivismq::commands::ProducerId::getConnectionId ()
const [virtual]

6.635.3.7 virtual std::string& activemq::commands::ProducerId::getConnectionId () [virtual]

6.635.3.8 virtual unsigned char ac-
tivismq::commands::ProducerId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

- 6.635.3.9 `const Pointer<SessionId>& activemq::commands::ProducerId::getParentId () const`
- 6.635.3.10 `virtual long long activemq::commands::ProducerId::getSessionId () const [virtual]`
- 6.635.3.11 `virtual long long activemq::commands::ProducerId::getValue () const [virtual]`
- 6.635.3.12 `virtual bool activemq::commands::ProducerId::operator< (const ProducerId & value) const [virtual]`
- 6.635.3.13 `ProducerId& activemq::commands::ProducerId::operator= (const ProducerId & other)`
- 6.635.3.14 `virtual bool activemq::commands::ProducerId::operator== (const ProducerId & value) const [virtual]`
- 6.635.3.15 `virtual void activemq::commands::ProducerId::setConnectionId (const std::string & connectionId) [virtual]`
- 6.635.3.16 `void activemq::commands::ProducerId::setProducerSessionKey (std::string sessionKey)`
- 6.635.3.17 `virtual void activemq::commands::ProducerId::setSessionId (long long sessionId) [virtual]`
- 6.635.3.18 `virtual void activemq::commands::ProducerId::setValue (long long value) [virtual]`
- 6.635.3.19 `virtual std::string activemq::commands::ProducerId::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.635.4 Field Documentation

- 6.635.4.1 `std::string activemq::commands::ProducerId::connectionId [protected]`
- 6.635.4.2 `const unsigned char activemq::commands::ProducerId::ID_ - PRODUCERID = 123 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.635.4.3 `long long activemq::commands::ProducerId::sessionId` [protected]

6.635.4.4 `long long activemq::commands::ProducerId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerId.h`

6.636 `activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2874).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller`:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.636.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2874). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.636.2 Constructor & Destructor Documentation

6.636.2.1 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::ProducerIdMarshaller () [inline]

6.636.2.2 virtual activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::~~ProducerIdMarshaller () [inline, virtual]

6.636.3 Member Function Documentation

6.636.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.636.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.636.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.636.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.636.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.636.3.6 virtual void **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal2**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.636.3.7 virtual void **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h`

6.637 **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2878).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller**:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.637.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2878). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.637.2 Constructor & Destructor Documentation

6.637.2.1 `activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

6.637.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

6.637.3 Member Function Documentation

6.637.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.637.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.637.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

```
6.637.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

```
6.637.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshall
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.637.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.637.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ProducerIdMarshaller.h**

6.638 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2881).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller`:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.638.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2881). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.638.2 Constructor & Destructor Documentation

6.638.2.1 `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

6.638.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

6.638.3 Member Function Documentation

6.638.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.638.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.638.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.638.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.638.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.638.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.638.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h`

6.639 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2885).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller**:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual \sim **ProducerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.639.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2885). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.639.2 Constructor & Destructor Documentation

6.639.2.1 `activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

6.639.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

6.639.3 Member Function Documentation

6.639.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.639.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.639.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.639.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.639.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.639.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.639.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h

6.640 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2889).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller**:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual \sim **ProducerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.640.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2889). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.640.2 Constructor & Destructor Documentation

6.640.2.1 `activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

6.640.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

6.640.3 Member Function Documentation

6.640.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.640.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.640.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.640.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.640.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.640.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.640.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarshaller.h

6.641 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2893).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual \sim **ProducerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.641.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2893). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.641.2 Constructor & Destructor Documentation

6.641.2.1 `activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

6.641.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

6.641.3 Member Function Documentation

6.641.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.641.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.641.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.641.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.641.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.641.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.641.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerIdMarshaller.h**

6.642 activemq::commands::ProducerInfo Class Reference

```
#include <src/main/activemq/commands/ProducerInfo.h>
```

Inheritance diagram for **activemq::commands::ProducerInfo**:

Public Member Functions

- **ProducerInfo** ()
- virtual **~ProducerInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **ProducerInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer< ProducerId > & getProducerId** () const
- virtual **Pointer< ProducerId > & getProducerId** ()
- virtual void **setProducerId** (const **Pointer< ProducerId > &producerId**)
- virtual const **Pointer< ActiveMQDestination > & getDestination** () const
- virtual **Pointer< ActiveMQDestination > & getDestination** ()
- virtual void **setDestination** (const **Pointer< ActiveMQDestination > &destination**)
- virtual const std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer< BrokerId > > &brokerPath**)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool **dispatchAsync**)
- virtual int **getWindowSize** () const
- virtual void **setWindowSize** (int **windowSize**)
- virtual bool **isProducerInfo** () const
- virtual **Pointer< Command > visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_PRODUCERINFO** = 6

Protected Attributes

- **Pointer< ProducerId > producerId**
- **Pointer< ActiveMQDestination > destination**
- std::vector< **decaf::lang::Pointer< BrokerId > > brokerPath**
- bool **dispatchAsync**
- int **windowSize**

6.642.1 Constructor & Destructor Documentation

6.642.1.1 `activemq::commands::ProducerInfo::ProducerInfo ()`

6.642.1.2 `virtual activemq::commands::ProducerInfo::~~ProducerInfo ()`
[virtual]

6.642.2 Member Function Documentation

6.642.2.1 `virtual ProducerInfo* activemq::commands::ProducerInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.642.2.2 `virtual void activemq::commands::ProducerInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.642.2.3 `Pointer<RemoveInfo> activemq::commands::ProducerInfo::createRemoveCommand () const`

6.642.2.4 `virtual bool activemq::commands::ProducerInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

- 6.642.2.5 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () const [virtual]`
- 6.642.2.6 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () [virtual]`
- 6.642.2.7 `virtual unsigned char activemq::commands::ProducerInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.642.2.8 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination () const [virtual]`
- 6.642.2.9 `virtual Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination () [virtual]`
- 6.642.2.10 `virtual Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId () [virtual]`
- 6.642.2.11 `virtual const Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId () const [virtual]`
- 6.642.2.12 `virtual int activemq::commands::ProducerInfo::getWindowSize () const [virtual]`
- 6.642.2.13 `virtual bool activemq::commands::ProducerInfo::isDispatchAsync () const [virtual]`
- 6.642.2.14 `virtual bool activemq::commands::ProducerInfo::isProducerInfo () const [inline, virtual]`

Returns

an answer of true to the **isProducerInfo()** (p. 2900) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 699).

- 6.642.2.15 `virtual void activemq::commands::ProducerInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath) [virtual]`
- 6.642.2.16 `virtual void activemq::commands::ProducerInfo::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.642.2.17 `virtual void activemq::commands::ProducerInfo::setDispatchAsync (bool dispatchAsync) [virtual]`
- 6.642.2.18 `virtual void activemq::commands::ProducerInfo::setProducerId (const Pointer< ProducerId > & producerId) [virtual]`
- 6.642.2.19 `virtual void activemq::commands::ProducerInfo::setWindowSize (int windowSize) [virtual]`
- 6.642.2.20 `virtual std::string activemq::commands::ProducerInfo::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

- 6.642.2.21 `virtual Pointer<Command> activemq::commands::ProducerInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1112).

6.642.3 Field Documentation

- 6.642.3.1 `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::ProducerInfo::brokerPath` [protected]
- 6.642.3.2 `Pointer<ActiveMQDestination>` `activemq::commands::ProducerInfo::destination`
[protected]
- 6.642.3.3 `bool` `activemq::commands::ProducerInfo::dispatchAsync` [protected]
- 6.642.3.4 `const unsigned char` `activemq::commands::ProducerInfo::ID_ - PRODUCERINFO = 6` [static]
- 6.642.3.5 `Pointer<ProducerId>` `activemq::commands::ProducerInfo::producerId`
[protected]
- 6.642.3.6 `int` `activemq::commands::ProducerInfo::windowSize` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerInfo.h`

6.643 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2902).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller`:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.643.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2902). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.643.2 Constructor & Destructor Documentation

6.643.2.1 `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::ProducerInfoMarshaller ()` [inline]

6.643.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::~~ProducerInfoMarshaller ()` [inline, virtual]

6.643.3 Member Function Documentation

6.643.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::createObject () const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.643.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::getDataStructureType () const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.643.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.643.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

6.643.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

```
6.643.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.643.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h`

6.644 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2906).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller`:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.644.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2906). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.644.2 Constructor & Destructor Documentation

6.644.2.1 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::ProducerInfoMarshaller () [inline]

6.644.2.2 virtual activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::~~ProducerInfoMarshaller () [inline, virtual]

6.644.3 Member Function Documentation

6.644.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.644.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.644.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.644.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

6.644.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

```
6.644.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

```
6.644.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h`

6.645 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2910).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller`:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.645.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2910). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.645.2 Constructor & Destructor Documentation

6.645.2.1 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::ProducerInfoMarshaller () [inline]

6.645.2.2 virtual
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::~~ProducerInfoMarshaller () [inline, virtual]

6.645.3 Member Function Documentation

6.645.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.645.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```

6.645.3.3 virtual void ac-
tivismq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

```

6.645.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

```

6.645.3.5 virtual int ac-
tivismq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

```
6.645.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

```
6.645.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h`

6.646 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2914).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller`:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.646.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2914). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.646.2 Constructor & Destructor Documentation

6.646.2.1 **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::ProducerInfoMarshaller**
() [inline]

6.646.2.2 **virtual**
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::~~ProducerInfoMarshaller
() [inline, virtual]

6.646.3 Member Function Documentation

6.646.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.646.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::getDataStructureType**
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```

6.646.3.3 virtual void ac-
tivismq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

```

6.646.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

```

6.646.3.5 virtual int ac-
tivismq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```
6.646.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.646.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h`

6.647 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2918).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller**:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.647.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2918). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.647.2 Constructor & Destructor Documentation

6.647.2.1 `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::ProducerInfoMarshaller () [inline]`

6.647.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::~~ProducerInfoMarshaller () [inline, virtual]`

6.647.3 Member Function Documentation

6.647.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.647.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```

6.647.3.3 virtual void ac-
tivismq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

```

6.647.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

```

6.647.3.5 virtual int ac-
tivismq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 704).

```
6.647.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 706).

```
6.647.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h`

6.648 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2922).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller**:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.648.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2922). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.648.2 Constructor & Destructor Documentation

6.648.2.1 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::ProducerInfoMarshaller () [inline]

6.648.2.2 virtual activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::~~ProducerInfoMarshaller () [inline, virtual]

6.648.3 Member Function Documentation

6.648.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.648.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.648.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.648.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

6.648.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 731).

```
6.648.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 732).

```
6.648.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h`

6.649 activemq::state::ProducerState Class Reference

```
#include <src/main/activemq/state/ProducerState.h>
```

Public Member Functions

- **ProducerState** (const **Pointer**< **ProducerInfo** > &info)
- virtual **~ProducerState** ()
- `std::string toString` () const
- const **Pointer**< **ProducerInfo** > & **getInfo** () const
- void **setTransactionState** (const **Pointer**< **TransactionState** > &transactionState)
- **Pointer**< **TransactionState** > **getTransactionState** () const

6.649.1 Constructor & Destructor Documentation

- 6.649.1.1 **activemq::state::ProducerState::ProducerState** (const **Pointer**< **ProducerInfo** > & *info*)
- 6.649.1.2 virtual **activemq::state::ProducerState::~~ProducerState** () [virtual]

6.649.2 Member Function Documentation

- 6.649.2.1 const **Pointer**<**ProducerInfo**>& **activemq::state::ProducerState::getInfo** () const [inline]
- 6.649.2.2 **Pointer**<**TransactionState**> **activemq::state::ProducerState::getTransactionState** () const
- 6.649.2.3 void **activemq::state::ProducerState::setTransactionState** (const **Pointer**< **TransactionState** > & *transactionState*)
- 6.649.2.4 `std::string` **activemq::state::ProducerState::toString** () const

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ProducerState.h`

6.650 decaf::util::Properties Class Reference

Java-like properties class for mapping string names to string values.

```
#include <src/main/decaf/util/Properties.h>
```

Public Member Functions

- **Properties** ()
- **Properties** (const **Properties** &src)
- virtual ~**Properties** ()
- **Properties** & **operator=** (const **Properties** &src)
Assignment Operator.
- bool **isEmpty** () const
Returns true if the properties object is empty.
- std::size_t **size** () const
- const char * **getProperty** (const std::string &name) const
Looks up the value for the given property.
- std::string **getProperty** (const std::string &name, const std::string &defaultValue) const
Looks up the value for the given property.
- std::string **setProperty** (const std::string &name, const std::string &value)
Sets the value for a given property.
- bool **hasProperty** (const std::string &name) const
Check to see if the Property exists in the set.
- std::string **remove** (const std::string &name)
Removes the property with the given name.
- std::vector< std::string > **propertyNames** () const
Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.
- std::vector< std::pair< std::string, std::string > > **toArray** () const
Method that serializes the contents of the property map to an array.
- void **copy** (const **Properties** &source)
*Copies the contents of the given properties object to this one, if the given **Properties** (p. 2927) instance in NULL then this **List** (p. 2190) is not modified.*
- **Properties** * **clone** () const
Clones this object.
- void **clear** ()
Clears all properties from the map.

- **bool equals** (const **Properties** &source) const
*Test whether two **Properties** (p. 2927) objects are equivalent.*
- **std::string toString** () const
*Formats the contents of the **Properties** (p. 2927) Object into a string that can be logged, etc.*
- **void load** (decaf::io::InputStream *stream) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)
Reads a property list (key and element pairs) from the input byte stream.
- **void load** (decaf::io::Reader *reader) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)
Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.
- **void store** (decaf::io::OutputStream *out, const std::string &comment) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
*Writes this property list (key and element pairs) in this **Properties** (p. 2927) table to the output stream in a format suitable for loading into a **Properties** (p. 2927) table using the load(InputStream) method.*
- **void store** (decaf::io::Writer *writer, const std::string &comments) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
*Writes this property list (key and element pairs) in this **Properties** (p. 2927) table to the output character stream in a format that can be read by the load(Reader) method.*

Protected Attributes

- **decaf::lang::Pointer< Properties > defaults**
Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

6.650.1 Detailed Description

Java-like properties class for mapping string names to string values. The **Properties** (p. 2927) list contains a key value pair of properties that can be loaded and stored to a stream. Each **Properties** (p. 2927) instance can contain an internal **Properties** (p. 2927) list that contains default values for keys not found in the **Properties** (p. 2927) **List** (p. 2190).

The **Properties** (p. 2927) list if a Thread Safe class, it can be shared amongst objects in multiple threads without the need for additional synchronization.

Since

1.0

6.650.2 Constructor & Destructor Documentation

6.650.2.1 decaf::util::Properties::Properties ()

6.650.2.2 decaf::util::Properties::Properties (const Properties & *src*)

6.650.2.3 virtual decaf::util::Properties::~~Properties () [virtual]

6.650.3 Member Function Documentation

6.650.3.1 void decaf::util::Properties::clear ()

Clears all properties from the map.

6.650.3.2 Properties* decaf::util::Properties::clone () const

Clones this object.

Returns

a replica of this object.

6.650.3.3 void decaf::util::Properties::copy (const Properties & *source*)

Copies the contents of the given properties object to this one, if the given **Properties** (p. 2927) instance is NULL then this **List** (p. 2190) is not modified.

Parameters

source The source properties object.

6.650.3.4 bool decaf::util::Properties::equals (const Properties & *source*) const

Test whether two **Properties** (p. 2927) objects are equivalent.

Two **Properties** (p. 2927) Objects are considered equivalent when they each contain the same number of elements and each key / value pair contained within the two are equal.

This comparison does not check the contents of the Defaults instance.

Parameters

source The **Properties** (p. 2927) object to compare this instance to.

Returns

true if the contents of the two **Properties** (p. 2927) objects are the same.

6.650.3.5 `const char* decaf::util::Properties::getProperty (const std::string & name) const`

Looks up the value for the given property.

Parameters

name The name of the property to be looked up.

Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

6.650.3.6 `std::string decaf::util::Properties::getProperty (const std::string & name, const std::string & defaultValue) const`

Looks up the value for the given property.

Parameters

name The name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

6.650.3.7 `bool decaf::util::Properties::hasProperty (const std::string & name) const`

Check to see if the Property exists in the set.

Parameters

name The property name to check for in this properties set.

Returns

true if property exists, false otherwise.

6.650.3.8 `bool decaf::util::Properties::isEmpty () const`

Returns true if the properties object is empty.

Returns

true if empty

6.650.3.9 void decaf::util::Properties::load (decaf::io::InputStream
* *stream*) throw (decaf::io::IOException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::NullPointerException)

Reads a property list (key and element pairs) from the input byte stream.

The input stream is in a simple line-oriented format as specified in load(Reader) and is assumed to use the ISO 8859-1 character encoding.

This method does not close the stream upon its return.

Parameters

stream The stream to read the properties data from.

Exceptions

IOException if there is an error while reading from the stream.

IllegalArgumentException if malformed data is found while reading the properties.

NullPointerException if the passed stream is Null.

6.650.3.10 void decaf::util::Properties::load (decaf::io::Reader
* *reader*) throw (decaf::io::IOException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::NullPointerException)

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.

Properties (p. 2927) are processed in terms of lines. There are two kinds of line, natural lines and logical lines. A natural line is defined as a line of characters that is terminated either by a set of line terminator characters (

or or

) or by the end of the stream. A natural line may be either a blank line, a comment line, or hold all or some of a key-element pair. A logical line holds all the data of a key-element pair, which may be spread out across several adjacent natural lines by escaping the line terminator sequence with a backslash character \. Note that a comment line cannot be extended in this manner; every natural line that is a comment must have its own comment indicator, as described below. Lines are read from input until the end of the stream is reached.

A natural line that contains only white space characters is considered blank and is ignored. A comment line has an ASCII '#' or '!' as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this format considers the characters space (' '), tab (""), and form feed ("") to be white space.

If a logical line is spread across several natural lines, the backslash escaping the line terminator sequence, the line terminator sequence, and any white space at the start of the following line have no effect on the key or element values. The remainder of the discussion of key and element parsing (when loading) will assume all the characters constituting the key and element appear on a single natural line after line continuation characters have been removed. Note that it is not sufficient to only examine the character preceding a line terminator sequence to decide if the line terminator is escaped; there must be an odd number of contiguous backslashes for the line terminator to be escaped. Since the input is processed from left to right, a non-zero even number of 2n contiguous backslashes before a line terminator (or elsewhere) encodes n backslashes after escape processing.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped '=', ':', or white space character other than a line terminator. All of these key termination characters may be included in the key by escaping them with a preceding backslash character; for example,

```
\:\=
```

would be the two-character key ":\=". Line terminator characters can be included using and

escape sequences. Any white space after the key is skipped; if the first non-white space character after the key is '=' or ':', then it is ignored and any white space characters after it are also skipped. All remaining characters on the line become part of the associated element string; if there are no remaining characters, the element is the empty string "". Once the raw character sequences constituting the key and element are identified, escape processing is performed as described above.

As an example, each of the following three lines specifies the key "Truth" and the associated element value "Beauty":

```
Truth = Beauty Truth:Beauty Truth :Beauty
```

As another example, the following three lines specify a single property:

```
fruits apple, banana, pear, \ cantaloupe, watermelon, \ kiwi, mango
```

The key is "fruits" and the associated element is: "apple, banana, pear, cantaloupe, watermelon, kiwi, mango"

Note that a space appears before each \ so that a space will appear after each comma in the final result; the \, line terminator, and leading white space on the continuation line are merely discarded and are not replaced by one or more other characters.

As a third example, the line:

```
cheeses
```

specifies that the key is "cheeses" and the associated element is the empty string "".

Characters in keys and elements can be represented in escape sequences similar to those used for character and string literals (see §3.3 and §3.10.6 of the Java Language Specification). The differences from the character escape sequences and Unicode escapes used for characters and strings are:

- Octal escapes are not recognized.
- The character sequence **does** not represent a backspace character.
- The method does not treat a backslash character, \, before a non-valid escape character as an error; the backslash is silently dropped. For example, in a C++ string the sequence "\z" would cause a compile time error. In contrast, this method silently drops the backslash. Therefore, this method treats the two character sequence "\b" as equivalent to the single character 'b'.
- Escapes are not necessary for single and double quotes; however, by the rule above, single and double quote characters preceded by a backslash still yield single and double quote characters, respectively.

This method does not close the Reader upon its return.

Parameters

reader The Reader that provides an character stream as input.

Exceptions

IOException if there is an error while reading from the stream.

IllegalArgumentException if malformed data is found while reading the properties.

NullPointerException if the passed stream is Null.

6.650.3.11 Properties& decaf::util::Properties::operator= (const Properties & src)

Assignment Operator.

Parameters

src The **Properties** (p. 2927) list to copy to this **List** (p. 2190).

Returns

a reference to this **List** (p. 2190) for use in chaining.

6.650.3.12 std::vector<std::string> decaf::util::Properties::propertyNames () const

Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.

Returns

a set of keys in this property list where the key and its corresponding value are strings, including the keys in the default property list.

6.650.3.13 std::string decaf::util::Properties::remove (const std::string & name)

Removes the property with the given name.

Parameters

name The name of the property to remove.

Returns

the previous value of the property if set, or empty string.

6.650.3.14 std::string decaf::util::Properties::setProperty (const std::string & name, const std::string & value)

Sets the value for a given property.

If the property already exists, overwrites the value.

Parameters

name The name of the value to be written.

value The value to be written.

Returns

the old value of the property or empty string if not set.

6.650.3.15 `std::size_t decaf::util::Properties::size () const`

Returns

The number of **Properties** (p. 2927) in this **Properties** (p. 2927) Object.

6.650.3.16 `void decaf::util::Properties::store (decaf::io::OutputStream * out,
const std::string & comment) throw (decaf::io::IOException,
decaf::lang::exceptions::NullPointerException)`

Writes this property list (key and element pairs) in this **Properties** (p.2927) table to the output stream in a format suitable for loading into a **Properties** (p.2927) table using the load(InputStream) method.

Properties (p. 2927) from the defaults table of this **Properties** (p. 2927) table (if any) are not written out by this method.

This method outputs the comments, properties keys and values in the same format as specified in store(Writer), with the following differences:

- The stream is written using the ISO 8859-1 character encoding.
- Characters not in Latin-1 in the comments are written as for their appropriate unicode hexadecimal value xxxx.
- Characters less than and characters greater than in property keys or values are written as for the appropriate hexadecimal value xxxx.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

Parameters

out The OutputStream instance to write the properties to.

comment A description of these properties that is written to the output stream.

Exceptions

IOException if there is an error while writing from the stream.

NullPointerException if the passed stream is Null.

6.650.3.17 `void decaf::util::Properties::store (decaf::io::Writer * writer,
const std::string & comments) throw (decaf::io::IOException,
decaf::lang::exceptions::NullPointerException)`

Writes this property list (key and element pairs) in this **Properties** (p. 2927) table to the output character stream in a format that can be read by the load(Reader) method.

Properties (p. 2927) from the defaults table of this **Properties** (p. 2927) table (if any) are not written out by this method.

If the comments argument is not empty, then an ASCII `#` character, the comments string, and a line separator are first written to the output stream. Thus, the comments can serve as an identifying comment. Any one of a line feed (`'`

`'`), a carriage return (`"`), or a carriage return followed immediately by a line feed in comments is replaced by a line separator generated by the Writer and if the next character in comments is not character `#` or character `!` then an ASCII `#` is written out after that line separator.

Next, a comment line is always written, consisting of an ASCII `#` character, the current date and time (as if produced by the `toString` method of **Date** (p. 1559) for the current time), and a line separator as generated by the Writer.

Then every entry in this **Properties** (p. 2927) table is written out, one per line. For each entry the key string is written, then an ASCII `=`, then the associated element string. For the key, all space characters are written with a preceding `\` character. For the element, leading space characters, but not embedded or trailing space characters, are written with a preceding `\` character. The key and element characters `#`, `!`, `=`, and `:` are written with a preceding backslash to ensure that they are properly loaded.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

Parameters

writer The Writer instance to use to output the properties.

comments A description of these properties that is written before writing the properties.

Exceptions

IOException if there is an error while writing from the stream.

NullPointerException if the passed stream is Null.

6.650.3.18 `std::vector< std::pair< std::string, std::string > >
decaf::util::Properties::toArray () const`

Method that serializes the contents of the property map to an array.

Returns

list of pairs where the first is the name and the second is the value.

6.650.3.19 `std::string decaf::util::Properties::toString () const`

Formats the contents of the **Properties** (p. 2927) Object into a string that can be logged, etc.

Returns

string value of this object.

6.650.4 Field Documentation

6.650.4.1 `decaf::lang::Pointer<Properties> decaf::util::Properties::defaults` [protected]

Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Properties.h`

6.651 `decaf::util::logging::PropertiesChangeListener` Class Reference

Defines the interface that classes can use to listen for change events on **Properties** (p. 2927).

```
#include <src/main/decaf/util/logging/PropertiesChangeListener.h>
```

Public Member Functions

- virtual `~PropertiesChangeListener ()`
- virtual void `onPropertiesReset ()=0`

*Indicates that the **Properties** (p. 2927) have all been reset and should be considered to be back to their default values.*

- virtual void `onPropertyChanged (const std::string &name, const std::string &oldValue, const std::string &newValue)=0`

Change Event, called when a property is changed, includes the name of the property that was changed along with it old and new values.

6.651.1 Detailed Description

Defines the interface that classes can use to listen for change events on **Properties** (p. 2927).

Since

1.0

6.651.2 Constructor & Destructor Documentation

6.651.2.1 virtual
 decaf::util::logging::PropertiesChangeListener::~~PropertiesChangeListener
 () [inline, virtual]

6.651.3 Member Function Documentation

6.651.3.1 virtual void de-
 caf::util::logging::PropertiesChangeListener::onPropertiesReset ()
 [pure virtual]

Indicates that the **Properties** (p.2927) have all been reset and should be considered to be back to their default values.

6.651.3.2 virtual void de-
 caf::util::logging::PropertiesChangeListener::onPropertyChanged (const
 std::string & *name*, const std::string & *oldValue*, const std::string &
newValue) [pure virtual]

Change Event, called when a property is changed, includes the name of the property that was changed along with it old and new values.

Parameters

- name* The name of the Property that changed.
- oldValue* The old Value of the Property.
- newValue* The new Value of the Property.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**PropertiesChangeListener.h**

6.652 decaf::net::ProtocolException Class Reference

```
#include <src/main/decaf/net/ProtocolException.h>
```

Inheritance diagram for decaf::net::ProtocolException:

Public Member Functions

- **ProtocolException** () throw ()
Default Constructor.
- **ProtocolException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **ProtocolException** (const **ProtocolException** &ex) throw ()

Copy Constructor.

- **ProtocolException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- **ProtocolException** (const std::exception *cause) throw ()

Constructor.

- **ProtocolException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **ProtocolException** * clone () const

Clones this exception.

- virtual ~**ProtocolException** () throw ()

6.652.1 Constructor & Destructor Documentation

6.652.1.1 decaf::net::ProtocolException::ProtocolException () throw () [inline]

Default Constructor.

6.652.1.2 decaf::net::ProtocolException::ProtocolException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.652.1.3 decaf::net::ProtocolException::ProtocolException (const ProtocolException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.652.1.4 decaf::net::ProtocolException::ProtocolException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.652.1.5 `decaf::net::ProtocolException::ProtocolException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.652.1.6 `decaf::net::ProtocolException::ProtocolException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.652.1.7 `virtual decaf::net::ProtocolException::~~ProtocolException () throw () [inline, virtual]`

6.652.2 Member Function Documentation

6.652.2.1 `virtual ProtocolException* decaf::net::ProtocolException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::IOException` (p. 2005).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ProtocolException.h`

6.653 decaf::security::PublicKey Class Reference

A public key.

```
#include <src/main/decaf/security/PublicKey.h>
```

Inheritance diagram for decaf::security::PublicKey:

Public Member Functions

- virtual `~PublicKey ()`

6.653.1 Detailed Description

A public key. This interface contains no methods or constants. It merely serves to group (and provide type safety for) all public key interfaces.

6.653.2 Constructor & Destructor Documentation

6.653.2.1 virtual decaf::security::PublicKey::~~PublicKey () [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/security/PublicKey.h`

6.654 decaf::io::PushbackInputStream Class Reference

A **PushbackInputStream** (p. 2940) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.

```
#include <src/main/decaf/io/PushbackInputStream.h>
```

Inheritance diagram for decaf::io::PushbackInputStream:

Public Member Functions

- **PushbackInputStream** (**InputStream** *stream, bool **own**=false)
*Creates a **PushbackInputStream** (p. 2940) and saves its argument, the input stream in, for later use.*
- **PushbackInputStream** (**InputStream** *stream, int bufSize, bool **own**=false) throw (decaf::lang::exceptions::IllegalArgumentException)
*Creates a **PushbackInputStream** (p. 2940) and saves its argument, the input stream in, for later use.*
- virtual `~PushbackInputStream ()`
- void **unread** (unsigned char value) throw (decaf::io::IOException)

Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.

- void **unread** (const unsigned char *buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

- void **unread** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

- virtual int **available** () const throw (decaf::io::IOException)

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

***IOException** (p. 2003) if an I/O error occurs.*

- virtual long long **skip** (long long num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Skips over and discards n bytes of data from this input stream.

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

***IOException** (p. 2003) if an I/O error occurs.*

***UnsupportedOperationException** if the concrete stream class does not support skipping bytes.*

- virtual void **mark** (int readLimit)

*Does nothing except throw an **IOException** (p. 2003).*

- virtual void **reset** () throw (decaf::io::IOException)

*Does nothing except throw an **IOException** (p. 2003).*

- virtual bool **markSupported** () const

*Does nothing except throw an **IOException** (p. 2003).*

Protected Member Functions

- virtual int **doReadByte** () throw (decaf::io::IOException)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsExpection, decaf::lang::exceptions::NullPointerException)

6.654.1 Detailed Description

A **PushbackInputStream** (p. 2940) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte. This is useful in situations where it is convenient for a fragment of code to read an indefinite number of data bytes that are delimited by a particular byte value; after reading the terminating byte, the code fragment can "unread" it, so that the next read operation on the input stream will reread the byte that was pushed back. For example, bytes representing the characters constituting an identifier might be terminated by a byte representing an operator character; a method whose job is to read just an identifier can read until it sees the operator and then push the operator back to be re-read.

Since

1.0

6.654.2 Constructor & Destructor Documentation

6.654.2.1 decaf::io::PushbackInputStream::PushbackInputStream (**InputStream** * *stream*, **bool** *own* = *false*)

Creates a **PushbackInputStream** (p. 2940) and saves its argument, the input stream in, for later use.

Initially, there is no pushed-back byte.

Parameters

stream The **InputStream** (p. 1909) instance to wrap.

Boolean value indicating if this **FilterInputStream** (p. 1771) owns the wrapped stream.

6.654.2.2 decaf::io::PushbackInputStream::PushbackInputStream (**InputStream** * *stream*, **int** *bufSize*, **bool** *own* = *false*) throw (decaf::lang::exceptions::IllegalArgumentException)

Creates a **PushbackInputStream** (p. 2940) and saves its argument, the input stream in, for later use.

Initially, there is no pushed-back byte.

Parameters

stream The **InputStream** (p. 1909) instance to wrap.

bufSize The number of byte to allocate for pushback into this stream.

Boolean value indicating if this **FilterInputStream** (p.1771) owns the wrapped stream.

Exceptions

IllegalArgumentException if the *bufSize* argument is < zero.

6.654.2.3 virtual decaf::io::PushbackInputStream::~~PushbackInputStream ()
[virtual]

6.654.3 Member Function Documentation

6.654.3.1 virtual int decaf::io::PushbackInputStream::available () const throw (decaf::io::IOException) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

Returns the sum of the number of pushed back bytes if any and the amount of bytes available in the underlying stream via a call to *available*.

Reimplemented from **decaf::io::FilterInputStream** (p.1773).

6.654.3.2 virtual int decaf::io::PushbackInputStream::doReadArrayBounded (unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p.1774).

6.654.3.3 virtual int decaf::io::PushbackInputStream::doReadByte () throw (decaf::io::IOException) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p.1774).

6.654.3.4 **virtual void decaf::io::PushbackInputStream::mark (int *readLimit*)** [virtual]

Does nothing except throw an **IOException** (p. 2003).

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit The max bytes read before marked position is invalid.

Reimplemented from **decaf::io::FilterInputStream** (p. 1775).

6.654.3.5 **virtual bool decaf::io::PushbackInputStream::markSupported () const** [inline, virtual]

Does nothing except throw an **IOException** (p. 2003).

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p. 1775).

6.654.3.6 **virtual void decaf::io::PushbackInputStream::reset () throw (decaf::io::IOException)** [virtual]

Does nothing except throw an **IOException** (p. 2003).

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2003) might be thrown. * If such an **IOException** (p. 2003) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 2003). * If an **IOException** (p. 2003) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that

will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2003).

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p. 1775).

6.654.3.7 **virtual long long decaf::io::PushbackInputStream::skip**
(**long long num**) **throw (decaf::io::IOException,**
decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

This method first skips bytes in the local pushed back buffer before attempting to complete the request by calling the underlying stream skip method with the remainder of bytes that needs to be skipped.

Reimplemented from **decaf::io::FilterInputStream** (p. 1776).

6.654.3.8 **void decaf::io::PushbackInputStream::unread (const unsigned**
char * buffer, int size) throw (decaf::io::IOException,
decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::NullPointerException)

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

Parameters

buffer The bytes to copy to the front of push back buffer.

size The size of the array to be copied.

Exceptions

NullPointerException if the buffer passed is NULL.

IndexOutOfBoundsException if the size value given is negative.

IOException (p. 2003) if there is not enough space in the pushback buffer or this stream has already been closed.

6.654.3.9 `void decaf::io::PushbackInputStream::unread (const unsigned char *
buffer, int size, int offset, int length) throw (decaf::io::IOException,
decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::NullPointerException)`

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

Parameters

buffer The bytes to copy to the front of push back buffer.

size The size of the array to be copied.

offset The position in the buffer to start copying from.

length The number of bytes to push back from the passed buffer.

Exceptions

NullPointerException if the buffer passed is NULL.

IndexOutOfBoundsException if the offset + length is greater than the buffer size.

IOException (p. 2003) if there is not enough space in the pushback buffer or this stream has already been closed.

6.654.3.10 `void decaf::io::PushbackInputStream::unread (unsigned char value)
throw (decaf::io::IOException)`

Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.

Parameters

value The byte that is to be placed at the front of the push back buffer.

Exceptions

IOException (p. 2003) if there is not enough space in the pushback buffer or this stream has already been closed.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/PushbackInputStream.h`

6.655 cms::Queue Class Reference

An interface encapsulating a provider-specific queue name.

```
#include <src/main/cms/Queue.h>
```

Inheritance diagram for cms::Queue:

Public Member Functions

- virtual `~Queue()`
- virtual `std::string getQueueName() const =0 throw (CMSEException)`
Gets the name of this queue.

6.655.1 Detailed Description

An interface encapsulating a provider-specific queue name. Messages sent to a **Queue** (p. 2947) are sent to a Single Subscriber on that **Queue** (p. 2947) **Destination** (p. 1610). This allows for Queues to be used as load balances implementing a SEDA based architecture. The length of time that a Provider will store a **Message** (p. 2375) in a **Queue** (p. 2947) is not defined by the CMS API, consult your Provider documentation for this information.

Since

1.0

6.655.2 Constructor & Destructor Documentation

6.655.2.1 virtual `cms::Queue::~Queue()` [inline, virtual]

6.655.3 Member Function Documentation

6.655.3.1 virtual `std::string cms::Queue::getQueueName() const throw (CMSEException)` [pure virtual]

Gets the name of this queue.

Returns

The queue name.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in `activemq::commands::ActiveMQQueue` (p. 438).

The documentation for this class was generated from the following file:

- `src/main/cms/Queue.h`

6.656 decaf::util::Queue< E > Class Template Reference

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

```
#include <src/main/decaf/util/Queue.h>
```

Inheritance diagram for decaf::util::Queue< E >:

Public Member Functions

- virtual `~Queue ()`
- virtual bool **offer** (const E &value)=0 throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Inserts the specified element into the queue provided that the condition allows such an operation.
- virtual bool **poll** (E &result)=0
Gets and removes the element in the head of the queue.
- virtual E **remove** ()=0 throw (decaf::lang::exceptions::NoSuchElementException)
Gets and removes the element in the head of the queue.
- virtual bool **peek** (E &result) const =0
Gets but not removes the element in the head of the queue.
- virtual E **element** () const =0 throw (decaf::lang::exceptions::NoSuchElementException)
Gets but not removes the element in the head of the queue.

6.656.1 Detailed Description

```
template<typename E> class decaf::util::Queue< E >
```

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection. Generally, a queue orders its elements by means of first-in-first-out. While priority queue orders its elements according to a comparator specified or the elements' natural order. Furthermore, a stack orders its elements last-in-first out.

Queue (p. 2948) does not provide blocking queue methods, which will block until the operation of the method is allowed. BlockingQueue interface defines such methods.

Unlike the Java **Queue** (p. 2948) interface the methods of this class cannot return null to indicate that a **Queue** (p. 2948) is empty since null has no meaning for elements such as classes, structs and primitive types and cannot be used in a meaningful way to check for an empty queue. Methods that would have returned null in the Java **Queue** (p. 2948) interface have been altered to return a boolean value indicating if the operation succeeded and take single argument that is a reference to the location where the returned value is to be assigned. This implies that elements in the **Queue** (p. 2948) must be *assignable* in order to utilize these methods.

Since

1.0

6.656.2 Constructor & Destructor Documentation

6.656.2.1 `template<typename E > virtual decaf::util::Queue< E >::~Queue ()`
[inline, virtual]

6.656.3 Member Function Documentation

6.656.3.1 `template<typename E > virtual E decaf::util::Queue< E >::element ()`
`const throw (decaf::lang::exceptions::NoSuchElementException)` [pure virtual]

Gets but not removes the element in the head of the queue.

Throws a `NoSuchElementException` if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException if there is no element in the queue.

Implemented in `decaf::util::AbstractQueue< E >` (p. 160).

6.656.3.2 `template<typename E > virtual bool decaf::util::Queue< E >::offer (`
`const E & value) throw (decaf::lang::exceptions::NullPointerException,`
`decaf::lang::exceptions::IllegalArgumentException)` [pure virtual]

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

Parameters

value the specified element to insert into the queue.

Returns

true if the operation succeeds and false if it fails.

Exceptions

NullPointerException if the **Queue** (p. 2948) implementation does not allow Null values to be inserted into the **Queue** (p. 2948).

IllegalArgumentException if some property of the specified element prevents it from being added to this queue

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3483), and `decaf::util::PriorityQueue< E >` (p. 2836).

Referenced by `decaf::util::AbstractQueue< E >::add()`.

6.656.3.3 `template<typename E> virtual bool decaf::util::Queue< E>::peek (E & result) const` [pure virtual]

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

Parameters

result Reference to an instance of the contained type to assigned the removed value to.

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in `decaf::util::PriorityQueue< E>` (p. 2837).

Referenced by `decaf::util::AbstractQueue< E>::element()`.

6.656.3.4 `template<typename E> virtual bool decaf::util::Queue< E>::poll (E & result)` [pure virtual]

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 2948) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

Parameters

result Reference to an instance of the contained type to assigned the removed value to.

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in `decaf::util::concurrent::SynchronousQueue< E>` (p. 3484), and `decaf::util::PriorityQueue< E>` (p. 2837).

Referenced by `decaf::util::AbstractQueue< E>::clear()`, and `decaf::util::AbstractQueue< E>::remove()`.

6.656.3.5 `template<typename E> virtual E decaf::util::Queue< E>::remove ()`
`throw (decaf::lang::exceptions::NoSuchElementException)` [pure virtual]

Gets and removes the element in the head of the queue.

Throws a `NoSuchElementException` if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException if there is no element in the queue.

Implemented in `decaf::util::AbstractQueue< E >` (p. 161), and `decaf::util::PriorityQueue< E >` (p. 2838).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Queue.h`

6.657 cms::QueueBrowser Class Reference

This class implements in interface for browsing the messages in a **Queue** (p. 2947) without removing them.

```
#include <src/main/cms/QueueBrowser.h>
```

Inheritance diagram for cms::QueueBrowser:

Public Member Functions

- virtual `~QueueBrowser()`
- virtual const **Queue** * `getQueue()` const =0 throw (cms::CMSEException)
- virtual std::string `getMessageSelector()` const =0 throw (cms::CMSEException)
- virtual **cms::MessageEnumeration** * `getEnumeration()` =0 throw (cms::CMSEException)

*Gets a pointer to an Enumeration object for browsing the Messages currently in the **Queue** (p. 2947) in the order that a client would receive them.*

6.657.1 Detailed Description

This class implements in interface for browsing the messages in a **Queue** (p. 2947) without removing them. To browse the contents of the **Queue** (p. 2947) the client calls the `getEnumeration` method to retrieve a new instance of a **Queue** (p. 2947) Enumerator. The client then calls the `hasMoreMessages` method of the Enumeration, if it returns true the client can safely call the `nextMessage` method of the Enumeration instance.

```
Enumeration* enumeration = queueBrowser->getEnumeration() (p. 2952);
```

```
while( enumeration->hasMoreMessages() ) { cms::Message (p. 2375)* message = enumeration->nextMessage();
```

```
// ... Do something with the Message (p. 2375).
```

```
delete message; }
```

Since

1.1

6.657.2 Constructor & Destructor Documentation

6.657.2.1 `virtual cms::QueueBrowser::~~QueueBrowser () [inline, virtual]`

6.657.3 Member Function Documentation

6.657.3.1 `virtual cms::MessageEnumeration* cms::QueueBrowser::getEnumeration () throw (cms::CMSEException) [pure virtual]`

Gets a pointer to an Enumeration object for browsing the Messages currently in the **Queue** (p. 2947) in the order that a client would receive them.

The pointer returned is owned by the browser and should not be deleted by the client application.

Returns

a pointer to a **Queue** (p. 2947) Enumeration, this Pointer is owned by the **QueueBrowser** (p. 2951) and should not be deleted by the client.

Exceptions

CMSEException (p. 1074) if an internal error occurs.

6.657.3.2 `virtual std::string cms::QueueBrowser::getMessageSelector () const throw (cms::CMSEException) [pure virtual]`

Returns

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

Exceptions

CMSEException (p. 1074) if an internal error occurs.

6.657.3.3 `virtual const Queue* cms::QueueBrowser::getQueue () const throw (cms::CMSEException) [pure virtual]`

Returns

the **Queue** (p. 2947) that this browser is listening on.

Exceptions

CMSEException (p. 1074) if an internal error occurs.

The documentation for this class was generated from the following file:

- `src/main/cms/QueueBrowser.h`

6.658 decaf::util::Random Class Reference

Random (p. 2953) Value Generator which is used to generate a stream of pseudorandom numbers.

```
#include <src/main/decaf/util/Random.h>
```

Inheritance diagram for decaf::util::Random:

Public Member Functions

- **Random** ()
Construct a random generator with the current time of day in milliseconds as the initial state.
- **Random** (unsigned long long seed)
Construct a random generator with the given `seed` as the initial state.
- bool **nextBoolean** ()
Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.
- double **nextDouble** ()
Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.
- float **nextFloat** ()
Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.
- double **nextGaussian** ()
Pseudo-randomly generates (approximately) a normally distributed `double` value with mean 0.0 and a standard deviation value of 1.0 using the polar method of G.
- int **nextInt** ()
Generates a uniformly distributed 32-bit `int` value from the this random number sequence.
- int **nextInt** (int n)
Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` (exclusively).
- long long **nextLong** ()
Generates a uniformly distributed 64-bit `int` value from the this random number sequence.
- virtual void **nextBytes** (std::vector< unsigned char > &buf)
Modifies the byte array by a random sequence of bytes generated by this random number generator.
- virtual void **nextBytes** (unsigned char *buf, int size)
Modifies the byte array by a random sequence of bytes generated by this random number generator.
- virtual void **setSeed** (unsigned long long seed)
*Modifies the seed using linear congruential formula presented in *The Art of Computer Programming*, Volume 2, Section 3.2.1.*

Protected Member Functions

- virtual int **next** (int bits)

*Answers a pseudo-random uniformly distributed **int** value of the number of bits specified by the argument **bits** as described by Donald E.*

6.658.1 Detailed Description

Random (p. 2953) Value Generator which is used to generate a stream of pseudorandom numbers. The algorithms implemented by class **Random** (p. 2953) use a protected utility method that on each invocation can supply up to 32 pseudorandomly generated bits.

Since

1.0

6.658.2 Constructor & Destructor Documentation

6.658.2.1 decaf::util::Random::Random ()

Construct a random generator with the current time of day in milliseconds as the initial state.

See also

setSeed (p. 2957)

6.658.2.2 decaf::util::Random::Random (unsigned long long seed)

Construct a random generator with the given **seed** as the initial state.

Parameters

seed the seed that will determine the initial state of this random number generator

See also

setSeed (p. 2957)

6.658.3 Member Function Documentation

6.658.3.1 virtual int decaf::util::Random::next (int bits) [protected, virtual]

Answers a pseudo-random uniformly distributed **int** value of the number of bits specified by the argument **bits** as described by Donald E.

Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

Returns

int a pseudo-random generated int number

Parameters

bits number of bits of the returned value

See also

nextBytes (p. 2955)
nextDouble (p. 2956)
nextFloat (p. 2956)
nextInt() (p. 2956)
nextInt(int) (p. 2957)
nextGaussian (p. 2956)
nextLong (p. 2957)

Reimplemented in **decaf::security::SecureRandom** (p. 3119).

6.658.3.2 bool decaf::util::Random::nextBoolean ()

Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.

Returns

boolean a pseudo-random, uniformly distributed boolean value

6.658.3.3 virtual void decaf::util::Random::nextBytes (unsigned char * buf, int size) [virtual]

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

buf non-null array to contain the new random bytes

See also

next (p. 2954)

Exceptions

NullPointerException if *buff* is NULL

IllegalArgumentException if *size* is negative

Reimplemented in **decaf::security::SecureRandom** (p. 3119).

6.658.3.4 virtual void decaf::util::Random::nextBytes (std::vector< unsigned char > & buf) [virtual]

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

buf non-null array to contain the new random bytes

See also

next (p. 2954)

Reimplemented in **decaf::security::SecureRandom** (p. 3120).

6.658.3.5 double decaf::util::Random::nextDouble ()

Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.

Returns

double

See also

nextFloat (p. 2956)

6.658.3.6 float decaf::util::Random::nextFloat ()

Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.

Returns

float a random float number between 0.0 and 1.0

See also

nextDouble (p. 2956)

6.658.3.7 double decaf::util::Random::nextGaussian ()

Pseudo-randomly generates (approximately) a normally distributed **double** value with mean 0.0 and a standard deviation value of 1.0 using the *polar method* of G.

E. P. Box, M. E. Muller, and G. Marsaglia, as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.4.1, subsection C, algorithm P

Returns

double

See also

nextDouble (p. 2956)

6.658.3.8 int decaf::util::Random::nextInt ()

Generates a uniformly distributed 32-bit **int** value from the this random number sequence.

Returns

int uniformly distributed **int** value

See also

next (p. 2954)

nextLong (p. 2957)

6.658.3.9 int decaf::util::Random::nextInt (int *n*)

Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of *n* (exclusively).

Parameters

n The int value that defines the max value of the return.

Returns

the next pseudo random int value.

Exceptions

IllegalArgumentException if *n* is less than or equal to zero.

6.658.3.10 long long decaf::util::Random::nextLong ()

Generates a uniformly distributed 64-bit int value from the this random number sequence.

Returns

64-bit int random number

See also

next (p. 2954)
nextInt() (p. 2956)
nextInt(int) (p. 2957)

6.658.3.11 virtual void decaf::util::Random::setSeed (unsigned long long *seed*)
[virtual]

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

Parameters

seed the seed that alters the state of the random number generator

See also

next (p. 2954)
Random() (p. 2954)
Random(long)

Reimplemented in **decaf::security::SecureRandom** (p. 3121).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Random.h**

6.659 decaf::lang::Readable Class Reference

A **Readable** (p. 2958) is a source of characters.

```
#include <src/main/decaf/lang/Readable.h>
```

Inheritance diagram for decaf::lang::Readable:

Public Member Functions

- virtual **~Readable** ()
- virtual int **read** (**decaf::nio::CharBuffer** *charBuffer)=0 throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException)

Attempts to read characters into the specified character buffer.

6.659.1 Detailed Description

A **Readable** (p. 2958) is a source of characters. Characters from a **Readable** (p. 2958) are made available to callers of the read method via a CharBuffer.

Since

1.0

6.659.2 Constructor & Destructor Documentation

6.659.2.1 virtual decaf::lang::Readable::~~Readable () [inline, virtual]

6.659.3 Member Function Documentation

6.659.3.1 virtual int decaf::lang::Readable::read (decaf::nio::CharBuffer * *charBuffer*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException) [pure virtual]

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

Parameters

charBuffer The Buffer to read Characters into.

Returns

The number of char values added to the buffer, or -1 if this source of characters is at its end

Exceptions

IOException - if an I/O error occurs

NullPointerException - if buffer is NULL.

ReadOnlyBufferException - if charBuffer is a read only buffer

Implemented in **decaf::io::Reader** (p. 2964).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Readable.h**

6.660 activemq::transport::inactivity::ReadChecker Class Reference

Runnable class that is used by the {.

```
#include <src/main/activemq/transport/inactivity/ReadChecker.h>
```

Inheritance diagram for activemq::transport::inactivity::ReadChecker:

Public Member Functions

- **ReadChecker** (**InactivityMonitor** *parent)
- virtual **~ReadChecker** ()
- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

6.660.1 Detailed Description

Runnable class that is used by the {.

See also

InactivityMonitor (p. 1874)} class the check for timeouts related to **transport** (p. 89) reads.

Since

3.1

6.660.2 Constructor & Destructor Documentation

6.660.2.1 `activemq::transport::inactivity::ReadChecker::ReadChecker (InactivityMonitor * parent)`

6.660.2.2 `virtual activemq::transport::inactivity::ReadChecker::~~ReadChecker ()` [virtual]

6.660.3 Member Function Documentation

6.660.3.1 `virtual void activemq::transport::inactivity::ReadChecker::run ()` [virtual]

Run method - called by the Thread class in the context of the thread.

Implements `decaf::lang::Runnable` (p. 3112).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/inactivity/ReadChecker.h`

6.661 decaf::io::Reader Class Reference

```
#include <src/main/decaf/io/Reader.h>
```

Inheritance diagram for `decaf::io::Reader`:

Public Member Functions

- virtual `~Reader ()`
- virtual void **mark** (int readAheadLimit) throw (decaf::io::IOException)
Marks the present position in the stream.
- virtual bool **markSupported** () const
*Tells whether this stream supports the **mark**() (p. 2962) operation.*
- virtual bool **ready** () const throw (decaf::io::IOException)
Tells whether this stream is ready to be read.
- virtual void **reset** () throw (decaf::io::IOException)
Resets the stream.
- virtual long long **skip** (long long count) throw (decaf::io::IOException)
Skips characters.
- virtual int **read** (std::vector< char > &buffer) throw (decaf::io::IOException)
Reads characters into an array.
- virtual int **read** (char *buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

Reads characters into an array, the method will attempt to read as much data as the size of the array.

- virtual int **read** (char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

Reads characters into a portion of an array.

- virtual int **read** () throw (decaf::io::IOException)

Reads a single character.

- virtual int **read** (decaf::nio::CharBuffer *charBuffer) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException)

Attempts to read characters into the specified character buffer.

Protected Member Functions

- **Reader** ()

- virtual int **doReadArrayBounded** (char *buffer, int size, int offset, int length)=0 throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Override this method to customize the functionality of the method read(unsigned char buffer, int size, int offset, int length).*

- virtual int **doReadVector** (std::vector< char > &buffer) throw (decaf::io::IOException)

Override this method to customize the functionality of the method read(std::vector<char>& buffer) (p. 2965).

- virtual int **doReadArray** (char *buffer, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

Override this method to customize the functionality of the method read(char buffer, std::size_t length).*

- virtual int **doReadChar** () throw (decaf::io::IOException)

Override this method to customize the functionality of the method read() (p. 2964).

- virtual int **doReadCharBuffer** (decaf::nio::CharBuffer *charBuffer) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException)

Override this method to customize the functionality of the method read(CharBuffer charBuffer).*

6.661.1 Constructor & Destructor Documentation

6.661.1.1 `decaf::io::Reader::Reader ()` [protected]

6.661.1.2 `virtual decaf::io::Reader::~~Reader ()` [virtual]

6.661.2 Member Function Documentation

6.661.2.1 `virtual int decaf::io::Reader::doReadArray (char *
buffer, int length) throw (decaf::io::IOException,
decaf::lang::exceptions::NullPointerException)` [protected, virtual]

Override this method to customize the functionality of the method `read(char* buffer, std::size_t length)`.

6.661.2.2 `virtual int decaf::io::Reader::doReadArrayBounded (char
* buffer, int size, int offset, int length) throw (decaf::io::IOException,
decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException)` [protected, pure virtual]

Override this method to customize the functionality of the method `read(unsigned char* buffer, int size, int offset, int length)`.

All subclasses must override this method to provide the basic **Reader** (p. 2960) functionality.

Implemented in **decaf::io::InputStreamReader** (p. 1921).

6.661.2.3 `virtual int decaf::io::Reader::doReadChar () throw (decaf::io::IOException)` [protected, virtual]

Override this method to customize the functionality of the method `read()` (p. 2964).

6.661.2.4 `virtual int decaf::io::Reader::doReadCharBuffer (decaf::nio::CharBuffer * charBuffer) throw (decaf::io::IOException,
decaf::lang::exceptions::NullPointerException,
decaf::nio::ReadOnlyBufferException)` [protected, virtual]

Override this method to customize the functionality of the method `read(CharBuffer* charBuffer)`.

6.661.2.5 `virtual int decaf::io::Reader::doReadVector (std::vector< char > & buffer) throw (decaf::io::IOException)` [protected, virtual]

Override this method to customize the functionality of the method `read(std::vector<char>& buffer)` (p. 2965).

6.661.2.6 `virtual void decaf::io::Reader::mark (int readAheadLimit) throw (decaf::io::IOException)` [virtual]

Marks the present position in the stream.

Subsequent calls to **reset()** (p. 2965) will attempt to reposition the stream to this point. Not all character-input streams support the **mark()** (p. 2962) operation.

Parameters

readAheadLimit Limit on the number of characters that may be read while still preserving the mark. After reading this many characters, attempting to reset the stream may fail.

Exceptions

IOException (p. 2003) if an I/O error occurs, or the stream does not support mark.

6.661.2.7 `virtual bool decaf::io::Reader::markSupported () const [inline, virtual]`

Tells whether this stream supports the **mark()** (p. 2962) operation.

The default implementation always returns false. Subclasses should override this method.

Returns

true if and only if this stream supports the mark operation.

6.661.2.8 `virtual int decaf::io::Reader::read (char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException) [virtual]`

Reads characters into a portion of an array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

Parameters

buffer The target char buffer.

size The size in bytes of the target buffer.

offset The position in the buffer to start filling.

length The maximum number of bytes to read.

Returns

The number of bytes read or -1 if the end of stream is reached.

Exceptions

IOException (p. 2003) thrown if an I/O error occurs.

NullPointerException if buffer is NULL.

IndexOutOfBoundsException if the offset + length is greater than the array size.

6.661.2.9 `virtual int decaf::io::Reader::read () throw (decaf::io::IOException)`
[virtual]

Reads a single character.

This method will block until a character is available, an I/O error occurs, or the end of the stream is reached.

Subclasses that intend to support efficient single-character input should override this method.

Returns

The character read, as an integer in the range 0 to 65535 (0x00-0xffff), or -1 if the end of the stream has been reached.

Exceptions

IOException (p. 2003) thrown if an I/O error occurs.

6.661.2.10 `virtual int decaf::io::Reader::read (char * buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)`
[virtual]

Reads characters into an array, the method will attempt to read as much data as the size of the array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

Parameters

buffer The target char buffer.

size The size in bytes of the target buffer.

Returns

The number of bytes read or -1 if the end of stream is reached.

Exceptions

IOException (p. 2003) thrown if an I/O error occurs.

NullPointerException if buffer is NULL.

6.661.2.11 `virtual int decaf::io::Reader::read (decaf::nio::CharBuffer * charBuffer) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException)` [virtual]

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

Parameters

charBuffer The Buffer to read Characters into.

Returns

The number of char values added to the buffer, or -1 if this source of characters is at its end

Exceptions

IOException (p. 2003) - if an I/O error occurs

NullPointerException - if buffer is NULL.

ReadOnlyBufferException - if charBuffer is a read only buffer

Implements **decaf::lang::Readable** (p. 2958).

6.661.2.12 `virtual int decaf::io::Reader::read (std::vector< char > & buffer)
throw (decaf::io::IOException) [virtual]`

Reads characters into an array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

Parameters

buffer The buffer to read characters into.

Returns

The number of characters read, or -1 if the end of the stream has been reached

Exceptions

IOException (p. 2003) thrown if an I/O error occurs.

6.661.2.13 `virtual bool decaf::io::Reader::ready () const throw (decaf::io::IOException) [virtual]`

Tells whether this stream is ready to be read.

Returns

True if the next **read()** (p. 2964) is guaranteed not to block for input, false otherwise. Note that returning false does not guarantee that the next read will block.

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented in **decaf::io::InputStreamReader** (p. 1921).

6.661.2.14 `virtual void decaf::io::Reader::reset () throw (decaf::io::IOException) [virtual]`

Resets the stream.

If the stream has been marked, then attempt to reposition it at the mark. If the stream has not been marked, then attempt to reset it in some way appropriate to the particular stream, for example by repositioning it to its starting point. Not all character-input streams support the **reset()** (p. 2965) operation, and some support **reset()** (p. 2965) without supporting **mark()** (p. 2962).

Exceptions

IOException (p. 2003) if an I/O error occurs.

6.661.2.15 `virtual long long decaf::io::Reader::skip (long long count) throw (decaf::io::IOException) [virtual]`

Skips characters.

This method will block until some characters are available, an I/O error occurs, or the end of the stream is reached.

Parameters

count The number of character to skip.

Returns

the number of Character actually skipped.

Exceptions

IOException (p. 2003) if an I/O error occurs.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Reader.h`

6.662 decaf::nio::ReadOnlyBufferException Class Reference

```
#include <src/main/decaf/nio/ReadOnlyBufferException.h>
```

Inheritance diagram for decaf::nio::ReadOnlyBufferException:

Public Member Functions

- **ReadOnlyBufferException** () throw ()
Default Constructor.
- **ReadOnlyBufferException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **ReadOnlyBufferException** (const ReadOnlyBufferException &ex) throw ()
Copy Constructor.

- **ReadOnlyBufferException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ReadOnlyBufferException** (const std::exception *cause) throw ()
Constructor.
- **ReadOnlyBufferException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **ReadOnlyBufferException * clone** () const
Clones this exception.
- virtual **~ReadOnlyBufferException** () throw ()

6.662.1 Constructor & Destructor Documentation

6.662.1.1 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException () throw () [inline]

Default Constructor.

6.662.1.2 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.662.1.3 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const ReadOnlyBufferException & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.662.1.4 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.662.1.5 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.662.1.6 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.662.1.7 `virtual decaf::nio::ReadOnlyBufferException::~~ReadOnlyBufferException () throw () [inline, virtual]`

6.662.2 Member Function Documentation

6.662.2.1 `virtual ReadOnlyBufferException* decaf::nio::ReadOnlyBufferException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from `decaf::lang::exceptions::UnsupportedOperationException` (p. 3659).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/ReadOnlyBufferException.h`

6.663 decaf::util::concurrent::locks::ReadWriteLock Class Reference

A **ReadWriteLock** (p. 2968) maintains a pair of associated locks, one for read-only operations and one for writing.

```
#include <src/main/decaf/util/concurrent/locks/ReadWriteLock.h>
```

Public Member Functions

- virtual **~ReadWriteLock** ()
- virtual **Lock & readLock** ()=0
Returns the lock used for reading.
- virtual **Lock & writeLock** ()=0
Returns the lock used for writing.

6.663.1 Detailed Description

A **ReadWriteLock** (p. 2968) maintains a pair of associated locks, one for read-only operations and one for writing. The read lock may be held simultaneously by multiple reader threads, so long as there are no writers. The write lock is exclusive.

All **ReadWriteLock** (p. 2968) implementations must guarantee that the memory synchronization effects of writeLock operations (as specified in the **Lock** (p. 2229) interface) also hold with respect to the associated readLock. That is, a thread successfully acquiring the read lock will see all updates made upon previous release of the write lock.

A read-write lock allows for a greater level of concurrency in accessing shared data than that permitted by a mutual exclusion lock. It exploits the fact that while only a single thread at a time (a writer thread) can modify the shared data, in many cases any number of threads can concurrently read the data (hence reader threads). In theory, the increase in concurrency permitted by the use of a read-write lock will lead to performance improvements over the use of a mutual exclusion lock. In practice this increase in concurrency will only be fully realized on a multi-processor, and then only if the access patterns for the shared data are suitable.

Whether or not a read-write lock will improve performance over the use of a mutual exclusion lock depends on the frequency that the data is read compared to being modified, the duration of the read and write operations, and the contention for the data - that is, the number of threads that will try to read or write the data at the same time. For example, a collection that is initially populated with data and thereafter infrequently modified, while being frequently searched (such as a directory of some kind) is an ideal candidate for the use of a read-write lock. However, if updates become frequent then the data spends most of its time being exclusively locked and there is little, if any increase in concurrency. Further, if the read operations are too short the overhead of the read-write lock implementation (which is inherently more complex than a mutual exclusion lock) can dominate the execution cost, particularly as many read-write lock implementations still serialize all threads through a small section of code. Ultimately, only profiling and measurement will establish whether the use of a read-write lock is suitable for your application.

Although the basic operation of a read-write lock is straight-forward, there are many policy decisions that an implementation must make, which may affect the effectiveness of the read-write lock in a given application. Examples of these policies include:

* Determining whether to grant the read lock or the write lock, when both readers and writers are waiting, at the time that a writer releases the write lock. Writer preference is common, as writes are expected to be short and infrequent. Reader preference is less common as it can lead to lengthy delays for a write if the readers are frequent and long-lived as expected. Fair, or "in-order" implementations are also possible. * Determining whether readers that request the read lock while a reader is active and a writer is waiting, are granted the read lock. Preference to the reader can delay the writer indefinitely, while preference to the writer can reduce the potential for concurrency. * Determining whether the locks are reentrant: can a thread with the write lock reacquire it? Can it acquire a read lock while holding the write lock? Is the read lock itself reentrant? * Can the write lock be downgraded to a read lock without allowing an intervening writer? Can a read lock be upgraded to a write lock, in preference to other waiting readers or writers?

You should consider all of these things when evaluating the suitability of a given implementation for your application.

Since

1.0

6.663.2 Constructor & Destructor Documentation

6.663.2.1 `virtual decaf::util::concurrent::locks::ReadWriteLock::~~ReadWriteLock () [inline, virtual]`

6.663.3 Member Function Documentation

6.663.3.1 `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::readLock () [pure virtual]`

Returns the lock used for reading.

Returns

the lock used for reading.

6.663.3.2 `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::writeLock () [pure virtual]`

Returns the lock used for writing.

Returns

the lock used for writing.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/ReadWriteLock.h`

6.664 activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::ReceiveExecutor:

Public Member Functions

- **ReceiveExecutor** (**CmsTemplate** *parent, **cms::Destination** *destination, const std::string &selector, bool noLocal)
- virtual ~**ReceiveExecutor** ()
- virtual void **doInCms** (**cms::Session** *session) throw (cms::CMSException)
Execute any number of operations against the supplied CMS session.
- virtual **cms::Destination** * **getDestination** (**cms::Session** *session AMQCPP_UNUSED) throw (cms::CMSException)
- **cms::Message** * **getMessage** ()

Protected Member Functions

- **ReceiveExecutor** (const **ReceiveExecutor** &)
- **ReceiveExecutor** & **operator=** (const **ReceiveExecutor** &)

Protected Attributes

- **cms::Destination** * destination
- std::string selector
- bool noLocal
- **cms::Message** * message
- **CmsTemplate** * parent

6.664.1 Constructor & Destructor Documentation

- 6.664.1.1** **activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor** (const **ReceiveExecutor** &) [inline, protected]
- 6.664.1.2** **activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor** (**CmsTemplate** * parent, **cms::Destination** * destination, const std::string & selector, bool noLocal) [inline]
- 6.664.1.3** virtual **activemq::cmsutil::CmsTemplate::ReceiveExecutor::~~ReceiveExecutor** () [inline, virtual]

6.664.2 Member Function Documentation

- 6.664.2.1** virtual void **activemq::cmsutil::CmsTemplate::ReceiveExecutor::doInCms** (**cms::Session** * session) throw (cms::CMSException) [virtual]

Execute any number of operations against the supplied CMS session.

Parameters

session the CMS Session

Exceptions

cms::CMSEException (p. 1074) if thrown by CMS API methods

Implements **activemq::cmsutil::SessionCallback** (p. 3161).

6.664.2.2 virtual **cms::Destination*** **activemq::cmsutil::CmsTemplate::ReceiveExecutor::getDestination** (**cms::Session *session** *AMQCPP_UNUSED*) throw (**cms::CMSEException**) [inline, virtual]

6.664.2.3 **cms::Message*** **activemq::cmsutil::CmsTemplate::ReceiveExecutor::getMessage** () [inline]

6.664.2.4 **ReceiveExecutor&** **activemq::cmsutil::CmsTemplate::ReceiveExecutor::operator=** (const **ReceiveExecutor &**) [inline, protected]

6.664.3 Field Documentation

6.664.3.1 **cms::Destination*** **activemq::cmsutil::CmsTemplate::ReceiveExecutor::destination** [protected]

6.664.3.2 **cms::Message*** **activemq::cmsutil::CmsTemplate::ReceiveExecutor::message** [protected]

6.664.3.3 **bool** **activemq::cmsutil::CmsTemplate::ReceiveExecutor::noLocal** [protected]

6.664.3.4 **CmsTemplate*** **activemq::cmsutil::CmsTemplate::ReceiveExecutor::parent** [protected]

6.664.3.5 **std::string** **activemq::cmsutil::CmsTemplate::ReceiveExecutor::selector** [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.665 activemq::core::RedeliveryPolicy Class Reference

Interface for a **RedeliveryPolicy** (p. 2972) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

```
#include <src/main/activemq/core/RedeliveryPolicy.h>
```

Inheritance diagram for activemq::core::RedeliveryPolicy:

Public Member Functions

- virtual `~RedeliveryPolicy ()`
- virtual double `getBackOffMultiplier ()` const =0
- virtual void `setBackOffMultiplier (double value)=0`
Sets the Back-Off Multiplier for Message Redelivery.
- virtual short `getCollisionAvoidancePercent ()` const =0
- virtual void `setCollisionAvoidancePercent (short value)=0`
- virtual long long `getInitialRedeliveryDelay ()` const =0
Gets the initial time that redelivery of messages is delayed.
- virtual void `setInitialRedeliveryDelay (long long value)=0`
Sets the initial time that redelivery will be delayed.
- virtual int `getMaximumRedeliveries ()` const =0
Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.
- virtual void `setMaximumRedeliveries (int maximumRedeliveries)=0`
Sets the Maximum allowable redeliveries for a Message.
- virtual long long `getRedeliveryDelay (long long previousDelay)=0`
Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.
- virtual bool `isUseCollisionAvoidance ()` const =0
- virtual void `setUseCollisionAvoidance (bool value)=0`
- virtual bool `isUseExponentialBackOff ()` const =0
- virtual void `setUseExponentialBackOff (bool value)=0`
- virtual `RedeliveryPolicy * clone ()` const =0
Create a copy of this Policy and return it.
- virtual void `configure (const decaf::util::Properties &properties)`
Checks the supplied properties object for properties matching the configurable settings of this class.

Static Public Attributes

- static const long long `NO__MAXIMUM__REDELIVERIES`

Protected Member Functions

- `RedeliveryPolicy ()`

6.665.1 Detailed Description

Interface for a **RedeliveryPolicy** (p. 2972) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

Since

3.2.0

6.665.2 Constructor & Destructor Documentation

6.665.2.1 `activemq::core::RedeliveryPolicy::RedeliveryPolicy ()` [protected]

6.665.2.2 `virtual activemq::core::RedeliveryPolicy::~~RedeliveryPolicy ()`
[virtual]

6.665.3 Member Function Documentation

6.665.3.1 `virtual RedeliveryPolicy* activemq::core::RedeliveryPolicy::clone ()`
`const` [pure virtual]

Create a copy of this Policy and return it.

Returns

pointer to a new **RedeliveryPolicy** (p. 2972) that is a copy of this one.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1570).

6.665.3.2 `virtual void activemq::core::RedeliveryPolicy::configure (const decaf::util::Properties & properties)` [virtual]

Checks the supplied properties object for properties matching the configurable settings of this class.

The default implementation looks for properties named with the prefix `cms.RedeliveryPolicy.XXX` where `XXX` is the name of a property with a public setter method. For instance `cms.RedeliveryPolicy.useExponentialBackOff` will be used to set the value of the use exponential back off toggle.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

Parameters

properties The Properties object used to configure this object.

Exceptions

NumberFormatException if a property that is numeric cannot be converted

IllegalArgumentException if a property can't be converted to the correct type.

6.665.3.3 `virtual double activemq::core::RedeliveryPolicy::getBackOffMultiplier () const` [pure virtual]

Returns

The value of the Back-Off Multiplier for Message Redelivery.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1571).

6.665.3.4 `virtual short activemq::core::RedeliveryPolicy::getCollisionAvoidancePercent () const` [pure virtual]

Returns

the currently set Collision Avoidance percentage.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1571).

6.665.3.5 `virtual long long activemq::core::RedeliveryPolicy::getInitialRedeliveryDelay () const` [pure virtual]

Gets the initial time that redelivery of messages is delayed.

Returns

the time in milliseconds that redelivery is delayed initially.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1571).

6.665.3.6 `virtual int activemq::core::RedeliveryPolicy::getMaximumRedeliveries () const` [pure virtual]

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

Returns

maximum allowed redeliveries for a message.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1571).

6.665.3.7 `virtual long long activemq::core::RedeliveryPolicy::getRedeliveryDelay (long long previousDelay)` [pure virtual]

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

Parameters

previousDelay The last delay that was used between message redeliveries.

Returns

the new delay to use before attempting another redelivery.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1572).

6.665.3.8 `virtual bool activemq::core::RedeliveryPolicy::isUseCollisionAvoidance () const` [pure virtual]

Returns

whether or not collision avoidance is enabled for this Policy.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1572).

6.665.3.9 `virtual bool activemq::core::RedeliveryPolicy::isUseExponentialBackOff () const` [pure virtual]

Returns

whether or not the exponential back off option is enabled.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1572).

6.665.3.10 `virtual void activemq::core::RedeliveryPolicy::setBackOffMultiplier (double value)` [pure virtual]

Sets the Back-Off Multiplier for Message Redelivery.

Parameters

value The new value for the back-off multiplier.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1572).

6.665.3.11 `virtual void activemq::core::RedeliveryPolicy::setCollisionAvoidancePercent (short value)` [pure virtual]

Parameters

value The collision avoidance percentage setting.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1572).

6.665.3.12 `virtual void activemq::core::RedeliveryPolicy::setInitialRedeliveryDelay (long long value)` [pure virtual]

Sets the initial time that redelivery will be delayed.

Parameters

value Time in Milliseconds to wait before starting redelivery.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1573).

6.665.3.13 `virtual void activemq::core::RedeliveryPolicy::setMaximumRedeliveries
(int maximumRedeliveries)` [pure virtual]

Sets the Maximum allowable redeliveries for a Message.

Parameters

maximumRedeliveries The maximum number of times that a message will be redelivered.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1573).

6.665.3.14 `virtual void activemq::core::RedeliveryPolicy::setUseCollisionAvoidance
(bool value)` [pure virtual]

Parameters

value Enable or Disable collision avoidance for this Policy.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1573).

6.665.3.15 `virtual void activemq::core::RedeliveryPolicy::setUseExponentialBackOff
(bool value)` [pure virtual]

Parameters

value Enable or Disable the exponential back off multiplier option.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1573).

6.665.4 Field Documentation

6.665.4.1 `const long long activemq::core::RedeliveryPolicy::NO_MAXIMUM_
REDELIVERIES` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/RedeliveryPolicy.h`

6.666 decaf::util::concurrent::locks::ReentrantLock Class Reference

A reentrant mutual exclusion **Lock** (p. 2229) with extended capabilities.

```
#include <src/main/decaf/util/concurrent/locks/ReentrantLock.h>
```

Inheritance diagram for `decaf::util::concurrent::locks::ReentrantLock`:

Public Member Functions

- **ReentrantLock** ()
- virtual **~ReentrantLock** ()
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Acquires the lock.
- virtual void **lockInterruptibly** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException)
Acquires the lock unless the current thread is interrupted.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Acquires the lock only if it is not held by another thread at the time of invocation.
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException)
Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Attempts to release this lock.
- virtual **Condition** * **newCondition** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException)
*Returns a **Condition** (p. 1156) instance for use with this **Lock** (p. 2229) instance.*
- int **getHoldCount** () const
Queries the number of holds on this lock by the current thread.
- bool **isHeldByCurrentThread** () const
Queries if this lock is held by the current thread.
- bool **isLocked** () const
Queries if this lock is held by any thread.
- bool **isFair** () const
Returns true if this lock has fairness set true.
- std::string **toString** () const
Returns a string identifying this lock, as well as its lock state.

6.666.1 Detailed Description

A reentrant mutual exclusion **Lock** (p. 2229) with extended capabilities. A **ReentrantLock** (p. 2977) is owned by the thread last successfully locking, but not yet unlocking it. A thread invoking lock will return, successfully acquiring the lock, when the lock is not owned by another thread. The method will return immediately if the current thread already owns the lock. This can be checked using methods **isHeldByCurrentThread**() (p. 2980), and **getHoldCount**() (p. 2979).

The constructor for this class accepts an optional fairness parameter. When set true, under contention, locks favor granting access to the longest-waiting thread. Otherwise this lock does not guarantee any particular access order. Programs using fair locks accessed by many threads may display lower overall throughput (i.e., are slower; often much slower) than those using the default setting, but have smaller variances in times to obtain locks and guarantee lack of starvation. Note however, that fairness of locks does not guarantee fairness of thread scheduling. Thus, one of many threads using a fair lock may obtain it multiple times in succession while other active threads are not progressing and not currently holding the lock. Also note that the untimed tryLock method does not honor the fairness setting. It will succeed if the lock is available even if other threads are waiting.

It is recommended practice to always immediately follow a call to lock with a try block, most typically in a before/after construction such as:

```
class X { private:  
    ReentrantLock (p. 2977) lock; // ...  
public:  
    void m() { lock.lock(); // block until condition holds  
    try { // ... method body } finally { lock.unlock() } } }
```

In addition to implementing the **Lock** (p. 2229) interface, this class defines methods isLocked and getLockQueueLength, as well as some associated protected access methods that may be useful for instrumentation and monitoring.

Since

1.0

6.666.2 Constructor & Destructor Documentation

6.666.2.1 decaf::util::concurrent::locks::ReentrantLock::ReentrantLock ()

6.666.2.2 virtual decaf::util::concurrent::locks::ReentrantLock::~~ReentrantLock () [virtual]

6.666.3 Member Function Documentation

6.666.3.1 int decaf::util::concurrent::locks::ReentrantLock::getHoldCount ()
const

Queries the number of holds on this lock by the current thread.

A thread has a hold on a lock for each lock action that is not matched by an unlock action.

The hold count information is typically only used for testing and debugging purposes. For example, if a certain section of code should not be entered with the lock already held then we can assert that fact:

```
class X { private:  
    ReentrantLock (p. 2977) lock; // ...  
public:  
    void m() { assert( lock.getHoldCount() == 0 ); lock.lock(); try { // ... method body } catch(...)  
    { lock.unlock(); } } }
```

Returns

the number of holds on this lock by the current thread, or zero if this lock is not held by the current thread

6.666.3.2 bool decaf::util::concurrent::locks::ReentrantLock::isFair () const

Returns true if this lock has fairness set true.

Returns

true if this lock has fairness set true

6.666.3.3 bool decaf::util::concurrent::locks::ReentrantLock::isHeldByCurrentThread () const

Queries if this lock is held by the current thread.

This method is typically used for debugging and testing. For example, a method that should only be called while a lock is held can assert that this is the case:

```
class X { private: ReentrantLock (p. 2977) lock = new ReentrantLock() (p. 2979); // ...
public: void m() { assert( lock.isHeldByCurrentThread() ); // ... method body } }
```

It can also be used to ensure that a reentrant lock is used in a non-reentrant manner, for example:

```
class X { private: ReentrantLock (p. 2977) lock = new ReentrantLock() (p. 2979); // ...
public: void m() { assert !lock.isHeldByCurrentThread() (p. 2980); lock.lock(); try { // ...
method body } finally { lock.unlock(); } } }
```

Returns

true if current thread holds this lock and false otherwise

6.666.3.4 bool decaf::util::concurrent::locks::ReentrantLock::isLocked () const

Queries if this lock is held by any thread.

This method is designed for use in monitoring of the system state, not for synchronization control.

Returns

true if any thread holds this lock and false otherwise

6.666.3.5 virtual void decaf::util::concurrent::locks::ReentrantLock::lock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Acquires the lock.

Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds the lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired, at which time the lock hold count is set to one.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

Implements **decaf::util::concurrent::locks::Lock** (p. 2231).

6.666.3.6 virtual void de-
caf::util::concurrent::locks::ReentrantLock::lockInterruptibly
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::InterruptedException) [virtual]

Acquires the lock unless the current thread is interrupted.

Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds this lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread.

If the lock is acquired by the current thread then the lock hold count is set to one.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implements **decaf::util::concurrent::locks::Lock** (p. 2231).

6.666.3.7 virtual Condition* de-
caf::util::concurrent::locks::ReentrantLock::newCondition
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Returns a **Condition** (p. 1156) instance for use with this **Lock** (p. 2229) instance.

The returned **Condition** (p. 1156) instance supports the same usages as do the **Mutex** (p. 2604) Class' methods (wait, notify, and notifyAll).

* If this lock is not held when any of the **Condition** (p. 1156) waiting or signalling methods are called, then an `IllegalMonitorStateException` is thrown. * When the condition waiting methods are called the lock is released and, before they return, the lock is reacquired and the lock hold count restored to what it was when the method was called. * If a thread is interrupted while waiting then the wait will terminate, an `InterruptedException` will be thrown, and the thread's interrupted status will be cleared. * Waiting threads are signaled in FIFO order. * The ordering of lock reacquisition for threads returning from waiting methods is the same as for threads initially acquiring the lock, which is in the default case not specified, but for fair locks favors those threads that have been waiting the longest.

Exceptions

RuntimeException if an error occurs while creating the **Condition** (p. 1156).

UnsupportedOperationException if this **Lock** (p. 2229) implementation does not support conditions

Implements **decaf::util::concurrent::locks::Lock** (p. 2232).

6.666.3.8 **std::string decaf::util::concurrent::locks::ReentrantLock::toString ()**
const

Returns a string identifying this lock, as well as its lock state.

The state, in brackets, includes either the String "Unlocked" or the String "Locked by" followed by the name of the owning thread.

Returns

a string identifying this lock, as well as its lock state

6.666.3.9 **virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock**
(long long *time*, const TimeUnit & *unit*)
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::InterruptedException) [virtual]

Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.

Acquires the lock if it is not held by another thread and returns immediately with the value true, setting the lock hold count to one. If this lock has been set to use a fair ordering policy then an available lock will not be acquired if any other threads are waiting for the lock. This is in contrast to the **tryLock()** (p. 2983) method. If you want a timed tryLock that does permit barging on a fair lock then combine the timed and un-timed forms together:

```
if (lock.tryLock() || lock.tryLock(timeout, unit) ) { ... }
```

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread;
or * The specified waiting time elapses

If the lock is acquired then the value true is returned and the lock hold count is set to one.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock, and over reporting the elapse of the waiting time.

Parameters

time the maximum time to wait for the lock

unit the time unit of the time argument

Returns

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

Exceptions

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implements **decaf::util::concurrent::locks::Lock** (p. 2233).

6.666.3.10 **virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock ()**
throw (decaf::lang::exceptions::RuntimeException) [virtual]

Acquires the lock only if it is not held by another thread at the time of invocation.

Acquires the lock if it is not held by another thread and returns immediately with the value true, setting the lock hold count to one. Even when this lock has been set to use a fair ordering policy, a call to **tryLock()** (p. 2983) will immediately acquire the lock if it is available, whether or not other threads are currently waiting for the lock. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting for this lock, then use **tryLock(0, TimeUnit.SECONDS)** (p. 3568) which is almost equivalent (it also detects interruption).

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then this method will return immediately with the value false.

Returns

true if the lock was acquired and false otherwise

Exceptions

RuntimeException if an error occurs while acquiring the lock.

Implements **decaf::util::concurrent::locks::Lock** (p. 2234).

6.666.3.11 **virtual void decaf::util::concurrent::locks::ReentrantLock::unlock**
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [virtual]

Attempts to release this lock.

If the current thread is the holder of this lock then the hold count is decremented. If the hold count is now zero then the lock is released. If the current thread is not the holder of this lock then **IllegalMonitorStateException** is thrown.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

Implements **decaf::util::concurrent::locks::Lock** (p. 2234).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/ReentrantLock.h`

6.667 decaf::util::concurrent::RejectedExecutionException Class Reference

```
#include <src/main/decaf/util/concurrent/RejectedExecutionException.h>
```

Inheritance diagram for **decaf::util::concurrent::RejectedExecutionException**:

Public Member Functions

- **RejectedExecutionException** () throw ()
Default Constructor.
- **RejectedExecutionException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **RejectedExecutionException** (const **RejectedExecutionException** &ex) throw ()
Copy Constructor.
- **RejectedExecutionException** (const std::exception *cause) throw ()
Constructor.
- **RejectedExecutionException** (const char *file, const int lineNumber, const char *msg,...)
 throw ()

Constructor - Initializes the file name and line number where this message occurred.

- **RejectedExecutionException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **RejectedExecutionException** * clone () const

Clones this exception.

- virtual ~**RejectedExecutionException** () throw ()

6.667.1 Constructor & Destructor Documentation

6.667.1.1 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException () throw () [inline]

Default Constructor.

6.667.1.2 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.667.1.3 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const RejectedExecutionException & ex) throw () [inline]

Copy Constructor.

Parameters

ex - The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

6.667.1.4 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const std::exception * cause) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.667.1.5 `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException`
 (`const char * file`, `const int lineNumber`, `const char * msg`, ...)
 throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs
lineNumber - The line number where the exception occurred.
msg - The message to report
 ... - list of primitives that are formatted into the message

6.667.1.6 `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException`
 (`const char * file`, `const int lineNumber`, `const std::exception * cause`, `const char * msg`, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs
lineNumber - The line number where the exception occurred.
cause - The exception that was the cause for this one to be thrown.
msg - The message to report
 ... - list of primitives that are formatted into the message

6.667.1.7 `virtual`
`decaf::util::concurrent::RejectedExecutionException::~~RejectedExecutionException`
 () throw () [inline, virtual]

6.667.2 Member Function Documentation

6.667.2.1 `virtual RejectedExecutionException* decaf::util::concurrent::RejectedExecutionException::clone` () `const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new instance this exception type with a copy the current state.

Reimplemented from `decaf::lang::Exception` (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/RejectedExecutionException.h`

6.668 decaf::util::concurrent::RejectedExecutionHandler Class Reference

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p.??).

```
#include <src/main/decaf/util/concurrent/RejectedExecutionHandler.h>
```

Public Member Functions

- virtual **~RejectedExecutionHandler** ()
- virtual void **rejectedExecution** (Runnable *r, ThreadPoolExecutor *executer)=0 throw (RejectedExecutionException)

*Method that may be invoked by a **ThreadPoolExecutor** (p.??) when **execute** (p.??) cannot accept a task.*

6.668.1 Detailed Description

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p.??).

Since

1.0

6.668.2 Constructor & Destructor Documentation

- 6.668.2.1** virtual
decaf::util::concurrent::RejectedExecutionHandler::~~RejectedExecutionHandler
() [inline, virtual]

6.668.3 Member Function Documentation

- 6.668.3.1** virtual void de-
caf::util::concurrent::RejectedExecutionHandler::rejectedExecution
(Runnable * r, ThreadPoolExecutor * *executer*) throw (RejectedExecutionException) [pure virtual]

Method that may be invoked by a **ThreadPoolExecutor** (p.??) when **execute** (p.??) cannot accept a task.

This may occur when no more threads or queue slots are available because their bounds would be exceeded, or upon shutdown of the **Executor** (p.1749).

In the absence of other alternatives, the method may throw an **RejectedExecutionException** (p.2984), which will be propagated to the caller of **execute** (p.??).

Parameters

- r* The pointer to the runnable task requested to be executed.
executer The pointer to the executor attempting to execute this task.

Exceptions

RejectedExecutionException (p.2984) if there is no remedy.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/RejectedExecutionHandler.h`

6.669 activemq::commands::RemoveInfo Class Reference

```
#include <src/main/activemq/commands/RemoveInfo.h>
```

Inheritance diagram for `activemq::commands::RemoveInfo`:

Public Member Functions

- **RemoveInfo** ()
- virtual **~RemoveInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **RemoveInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **DataStructure** > & **getObjectId** () const
- virtual **Pointer**< **DataStructure** > & **getObjectId** ()
- virtual void **setObjectId** (const **Pointer**< **DataStructure** > &objectId)
- virtual long long **getLastDeliveredSequenceId** () const
- virtual void **setLastDeliveredSequenceId** (long long lastDeliveredSequenceId)
- virtual bool **isRemoveInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REMOVEINFO** = 12

Protected Attributes

- `Pointer< DataStructure > objectId`
- `long long lastDeliveredSequenceId`

6.669.1 Constructor & Destructor Documentation

6.669.1.1 `activemq::commands::RemoveInfo::RemoveInfo ()`

6.669.1.2 `virtual activemq::commands::RemoveInfo::~~RemoveInfo () [virtual]`

6.669.2 Member Function Documentation

6.669.2.1 `virtual RemoveInfo* activemq::commands::RemoveInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.669.2.2 `virtual void activemq::commands::RemoveInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.669.2.3 `virtual bool activemq::commands::RemoveInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.669.2.4 `virtual unsigned char activemq::commands::RemoveInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSetructure** (p. 1553) type copy.

Implements **activemq::commands::DataSetructure** (p. 1557).

6.669.2.5 `virtual long long activemq::commands::RemoveInfo::getLastDeliveredSequenceId () const [virtual]`

6.669.2.6 `virtual const Pointer<DataSetructure>& activemq::commands::RemoveInfo::getObjectId () const [virtual]`

6.669.2.7 `virtual Pointer<DataSetructure>& activemq::commands::RemoveInfo::getObjectId () [virtual]`

6.669.2.8 `virtual bool activemq::commands::RemoveInfo::isRemoveInfo () const [inline, virtual]`

Returns

an answer of true to the **isRemoveInfo()** (p. 2990) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 699).

6.669.2.9 `virtual void activemq::commands::RemoveInfo::setLastDeliveredSequenceId (long long lastDeliveredSequenceId) [virtual]`

6.669.2.10 `virtual void activemq::commands::RemoveInfo::setObjectId (const Pointer< DataSetructure > & objectId) [virtual]`

6.669.2.11 `virtual std::string activemq::commands::RemoveInfo::toString () const [virtual]`

Returns a string containing the information for this **DataSetructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

6.669.2.12 `virtual Pointer<Command> activemq::commands::RemoveInfo::visit
 (activemq::state::CommandVisitor * visitor) throw (
 exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.669.3 Field Documentation

6.669.3.1 `const unsigned char activemq::commands::RemoveInfo::ID _ -
 REMOVEINFO = 12 [static]`

6.669.3.2 `long long activemq::commands::RemoveInfo::lastDeliveredSequenceId
 [protected]`

6.669.3.3 `Pointer<DataStructure> activemq::commands::RemoveInfo::objectId
 [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/RemoveInfo.h`

6.670 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2991).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller`:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.670.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2991). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.670.2 Constructor & Destructor Documentation

6.670.2.1 **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::RemoveInfoMarshaller**
() [inline]

6.670.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::~~RemoveInfoMarshaller
() [inline, virtual]

6.670.3 Member Function Documentation

6.670.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.670.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::getDataSetType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.670.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataSet * dataSet, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 736).

6.670.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataSet * dataSet, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 737).

6.670.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

6.670.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

6.670.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h`

6.671 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2995).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller**:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.671.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2995). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.671.2 Constructor & Destructor Documentation

6.671.2.1 **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::RemoveInfoMarshaller** () [inline]

6.671.2.2 **virtual**
activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::~~RemoveInfoMarshaller
() [inline, virtual]

6.671.3 Member Function Documentation

6.671.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.671.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.671.3.3 virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 729).

6.671.3.4 virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 730).

6.671.3.5 virtual int activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

```
6.671.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

```
6.671.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h`

6.672 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2999).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller**:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.672.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2999). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.672.2 Constructor & Destructor Documentation

6.672.2.1 `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::RemoveInfoMarshaller () [inline]`

6.672.2.2 `virtual activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::~~RemoveInfoMarshaller () [inline, virtual]`

6.672.3 Member Function Documentation

6.672.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.672.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.672.3.3 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.672.3.4 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.672.3.5 virtual int activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

```
6.672.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.672.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h`

6.673 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3003).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller**:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.673.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3003). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.673.2 Constructor & Destructor Documentation

6.673.2.1 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::RemoveInfoMarshaller () [inline]

6.673.2.2 virtual
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::~~RemoveInfoMarshaller () [inline, virtual]

6.673.3 Member Function Documentation

6.673.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.673.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.673.3.3 virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.673.3.4 virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.673.3.5 virtual int activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

```
6.673.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

```
6.673.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h`

6.674 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3007).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller**:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.674.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p.3007). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.674.2 Constructor & Destructor Documentation

6.674.2.1 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::RemoveInfoMarshaller () [inline]

6.674.2.2 virtual
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::~~RemoveInfoMarshaller () [inline, virtual]

6.674.3 Member Function Documentation

6.674.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.674.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.674.3.3 virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.674.3.4 virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

6.674.3.5 virtual int activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```
6.674.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.674.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h`

6.675 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3011).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller**:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.675.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p.3011). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.675.2 Constructor & Destructor Documentation

6.675.2.1 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::RemoveInfoMarshaller () [inline]

6.675.2.2 virtual
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::~~RemoveInfoMarshaller () [inline, virtual]

6.675.3 Member Function Documentation

6.675.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.675.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.675.3.3 virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 709).

6.675.3.4 virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 710).

6.675.3.5 virtual int activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

```
6.675.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.675.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h`

6.676 activemq::commands::RemoveSubscriptionInfo Class Reference

```
#include <src/main/activemq/commands/RemoveSubscriptionInfo.h>
```

Inheritance diagram for **activemq::commands::RemoveSubscriptionInfo**:

Public Member Functions

- **RemoveSubscriptionInfo** ()
- virtual **~RemoveSubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaller share.
- virtual **RemoveSubscriptionInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)

- virtual bool **isRemoveSubscriptionInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REMOVESUBSCRIPTIONINFO** = 9

Protected Attributes

- **Pointer< ConnectionId > connectionId**
- std::string **subscriptionName**
- std::string **clientId**

6.676.1 Constructor & Destructor Documentation

6.676.1.1 **activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo** ()

6.676.1.2 **virtual**
activemq::commands::RemoveSubscriptionInfo::~~RemoveSubscriptionInfo
() [virtual]

6.676.2 Member Function Documentation

6.676.2.1 **virtual RemoveSubscriptionInfo* ac-**
tivemq::commands::RemoveSubscriptionInfo::cloneDataStructure ()
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.676.2.2 **virtual void ac-**
tivemq::commands::RemoveSubscriptionInfo::copyDataStructure (const
DataStructure * src) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.676.2.3 `virtual bool activemq::commands::RemoveSubscriptionInfo::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.676.2.4 `virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () const [virtual]`

6.676.2.5 `virtual std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () [virtual]`

6.676.2.6 `virtual const Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () const [virtual]`

6.676.2.7 `virtual Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () [virtual]`

6.676.2.8 `virtual unsigned char activemq::commands::RemoveSubscriptionInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.676.2.9** `virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName () const [virtual]`
- 6.676.2.10** `virtual std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName () [virtual]`
- 6.676.2.11** `virtual bool activemq::commands::RemoveSubscriptionInfo::isRemoveSubscriptionInfo () const [inline, virtual]`

Returns

an answer of true to the `isRemoveSubscriptionInfo()` (p. 3018) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 699).

- 6.676.2.12** `virtual void activemq::commands::RemoveSubscriptionInfo::setClientId (const std::string & clientId) [virtual]`
- 6.676.2.13** `virtual void activemq::commands::RemoveSubscriptionInfo::setConnectionId (const Pointer< ConnectionId > & connectionId) [virtual]`
- 6.676.2.14** `virtual void activemq::commands::RemoveSubscriptionInfo::setSubscriptionName (const std::string & subscriptionName) [virtual]`
- 6.676.2.15** `virtual std::string activemq::commands::RemoveSubscriptionInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 700).

- 6.676.2.16** `virtual Pointer<Command> activemq::commands::RemoveSubscriptionInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

6.677

activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller

Class Reference

3023

Implements **activemq::commands::Command** (p. 1112).

6.676.3 Field Documentation

6.676.3.1 `std::string activemq::commands::RemoveSubscriptionInfo::clientId`
[protected]

6.676.3.2 `Pointer<ConnectionId> activemq::commands::RemoveSubscriptionInfo::connectionId`
[protected]

6.676.3.3 `const unsigned char activemq::commands::RemoveSubscriptionInfo::ID_REMOVE_SUBSCRIPTION_INFO = 9` [static]

6.676.3.4 `std::string activemq::commands::RemoveSubscriptionInfo::subscriptionName`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/RemoveSubscriptionInfo.h`

6.677 **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3019).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.677.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3019).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.677.2 Constructor & Destructor Documentation

6.677.2.1 **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller** () [inline]

6.677.2.2 virtual **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller** () [inline, virtual]

6.677.3 Member Function Documentation

6.677.3.1 virtual **commands::DataStructure*** **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.677

activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller

Class Reference

3025

```
6.677.3.2 virtual unsigned char ac-
          tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::getDataSt
          ( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```
6.677.3.3 virtual void ac-
          tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseMars
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
            decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 716).

```
6.677.3.4 virtual void ac-
          tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseUnm
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, decaf::io::DataInputStream * dataIn ) throw (
            decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 717).

6.677.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

6.677.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

6.677.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h`

6.678 `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `RemoveSubscriptionInfoMarshaller` (p. 3023).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller`:

Public Member Functions

- `RemoveSubscriptionInfoMarshaller ()`
- `virtual ~RemoveSubscriptionInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.678.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3023).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.678.2 Constructor & Destructor Documentation

6.678.2.1 **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller** () [inline]

6.678.2.2 **virtual**
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller () [inline, virtual]

6.678.3 Member Function Documentation

6.678.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.678.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

6.678

activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller

Class Reference

3029

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.678.3.3 virtual void **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseMarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - **BinaryWriter** that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 702).

6.678.3.4 virtual void **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - **BinaryReader** that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 703).

6.678.3.5 virtual int **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

```
6.678.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.678.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightUnm
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

6.679

activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller

Class Reference

3031

bs - `BooleanStream` stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h`

6.679 **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3027).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.679.1 Detailed Description

Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 3027).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.679.2 Constructor & Destructor Documentation

6.679.2.1 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller () [inline]

6.679.2.2 virtual
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller () [inline, virtual]

6.679.3 Member Function Documentation

6.679.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements activemq::wireformat::openwire::marshal::DataStreamMarshaller (p. 1505).

6.679.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements activemq::wireformat::openwire::marshal::DataStreamMarshaller (p. 1511).

6.679

activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller

Class Reference

3033

```
6.679.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

```
6.679.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseUnm
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

```
6.679.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

```
6.679.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

```
6.679.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h`

6.680 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3031).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller**:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.680.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3031).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.680.2 Constructor & Destructor Documentation

6.680.2.1 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller () [inline]

6.680.2.2 virtual
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller () [inline, virtual]

6.680.3 Member Function Documentation

6.680.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.680.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.680

activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller

Class Reference

3037

```
6.680.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::looseMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

```
6.680.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::looseUnm
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

```
6.680.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

```
6.680.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

```
6.680.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscriptionInfoMarshaller.h`

6.681 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3035).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller**:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.681.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3035).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.681.2 Constructor & Destructor Documentation

6.681.2.1 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller () [inline]

6.681.2.2 virtual
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller () [inline, virtual]

6.681.3 Member Function Documentation

6.681.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.681.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.681

activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller

Class Reference

3041

```
6.681.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::looseMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

```
6.681.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::looseUnm
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

```
6.681.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```
6.681.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.681.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h`

6.682 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3039).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.682.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3039).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.682.2 Constructor & Destructor Documentation

6.682.2.1 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller () [inline]

6.682.2.2 virtual
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller () [inline, virtual]

6.682.3 Member Function Documentation

6.682.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.682.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.682

activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller

Class Reference

3045

```
6.682.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::looseMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

```
6.682.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::looseUnm
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

```
6.682.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

```
6.682.3.6  virtual void ac-
            tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMars
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.682.3.7  virtual void ac-
            tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightUnma
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataInputStream * dataIn,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h`

6.683 activemq::commands::ReplayCommand Class Reference

```
#include <src/main/activemq/commands/ReplayCommand.h>
```

Inheritance diagram for **activemq::commands::ReplayCommand**:

Public Member Functions

- **ReplayCommand** ()
- virtual **~ReplayCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaller share.
- virtual **ReplayCommand * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual int **getFirstNakNumber** () const
- virtual void **setFirstNakNumber** (int firstNakNumber)
- virtual int **getLastNakNumber** () const
- virtual void **setLastNakNumber** (int lastNakNumber)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REPLAYCOMMAND** = 65

Protected Attributes

- int **firstNakNumber**
- int **lastNakNumber**

6.683.1 Constructor & Destructor Documentation

6.683.1.1 `activemq::commands::ReplayCommand::ReplayCommand ()`

6.683.1.2 `virtual activemq::commands::ReplayCommand::~~ReplayCommand ()`
[virtual]

6.683.2 Member Function Documentation

6.683.2.1 `virtual ReplayCommand* activemq::commands::ReplayCommand::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.683.2.2 `virtual void activemq::commands::ReplayCommand::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.683.2.3 `virtual bool activemq::commands::ReplayCommand::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.683.2.4 `virtual unsigned char activemq::commands::ReplayCommand::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

6.683.2.5 `virtual int activemq::commands::ReplayCommand::getFirstNakNumber () const [virtual]`

6.683.2.6 `virtual int activemq::commands::ReplayCommand::getLastNakNumber () const [virtual]`

6.683.2.7 `virtual void activemq::commands::ReplayCommand::setFirstNakNumber (int firstNakNumber) [virtual]`

6.683.2.8 `virtual void activemq::commands::ReplayCommand::setLastNakNumber (int lastNakNumber) [virtual]`

6.683.2.9 `virtual std::string activemq::commands::ReplayCommand::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

6.683.2.10 `virtual Pointer<Command> activemq::commands::ReplayCommand::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.683.3 Field Documentation

6.683.3.1 `int activemq::commands::ReplayCommand::firstNakNumber` [protected]

6.683.3.2 `const unsigned char activemq::commands::ReplayCommand::ID` - `REPLAYCOMMAND = 65` [static]

6.683.3.3 `int activemq::commands::ReplayCommand::lastNakNumber` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ReplayCommand.h`

6.684 `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3046).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual `~ReplayCommandMarshaller` ()
- virtual `commands::DataStructure * createObject ()` const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.684.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3046). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.684.2 Constructor & Destructor Documentation

6.684.2.1 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::ReplayCommandMarshaller () [inline]

6.684.2.2 virtual activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::~~ReplayCommandMarshaller () [inline, virtual]

6.684.3 Member Function Documentation

6.684.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.684.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.684.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.684.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

6.684.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 711).

```
6.684.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 712).

```
6.684.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h`

6.685 activemq::wireformat::openwire::marshal::v1::ReplayCommandMa Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3050).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller**:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.685.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p.3050). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.685.2 Constructor & Destructor Documentation

6.685.2.1 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::ReplayCommandMarshaller () [inline]

6.685.2.2 virtual
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::~~ReplayCommandMarshaller () [inline, virtual]

6.685.3 Member Function Documentation

6.685.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.685.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.685.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.685.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.685.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 718).

```
6.685.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 719).

```
6.685.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h`

6.686 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3054).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.686.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p.3054). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.686.2 Constructor & Destructor Documentation

6.686.2.1 **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::ReplayCommandMarshaller**
() [inline]

6.686.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::~~ReplayCommandMarshaller
() [inline, virtual]

6.686.3 Member Function Documentation

6.686.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.686.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::getDataStructureType**
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.686.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.686.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

6.686.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 738).

```
6.686.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 739).

```
6.686.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h`

6.687 activemq::wireformat::openwire::marshal::v3::ReplayCommandMa Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3058).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.687.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p.3058). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.687.2 Constructor & Destructor Documentation

6.687.2.1 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::ReplayCommandMarshaller () [inline]

6.687.2.2 virtual
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::~~ReplayCommandMarshaller () [inline, virtual]

6.687.3 Member Function Documentation

6.687.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.687.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.687.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.687.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.687.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 704).

```
6.687.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 706).

```
6.687.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h`

6.688 activemq::wireformat::openwire::marshal::v6::ReplayCommandMa Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3062).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller**:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.688.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p.3062). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.688.2 Constructor & Destructor Documentation

6.688.2.1 activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::ReplayCommandMarshaller () [inline]

6.688.2.2 virtual activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::~~ReplayCommandMarshaller () [inline, virtual]

6.688.3 Member Function Documentation

6.688.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.688.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.688.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.688.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

6.688.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 731).

```
6.688.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 732).

```
6.688.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ReplayCommandMarshaller.h`

6.689 activemq::wireformat::openwire::marshal::v5::ReplayCommandMa Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3066).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller**:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.689.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p.3066). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.689.2 Constructor & Destructor Documentation

6.689.2.1 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::ReplayCommandMarshaller () [inline]

6.689.2.2 virtual
 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::~~ReplayCommandMarshaller () [inline, virtual]

6.689.3 Member Function Documentation

6.689.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.689.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.689.3.3 virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

6.689.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

6.689.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```
6.689.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.689.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h`

6.690 **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor** Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor**:

Public Member Functions

- **ResolveProducerExecutor** (**ProducerCallback** *action, **CmsTemplate** *parent, const std::string &destinationName)
- virtual ~**ResolveProducerExecutor** ()
- virtual **cms::Destination** * **getDestination** (**cms::Session** *session) throw (**cms::CMSException**)

Protected Member Functions

- **ResolveProducerExecutor** & **operator=** (const **ResolveProducerExecutor** &)

6.690.1 Constructor & Destructor Documentation

- 6.690.1.1** `activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::ResolveProducerExecutor (ProducerCallback * action, CmsTemplate * parent, const std::string & destinationName)` [inline]
- 6.690.1.2** `virtual activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::~~ResolveProducerExecutor ()` [inline, virtual]

6.690.2 Member Function Documentation

- 6.690.2.1** `virtual cms::Destination* activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::getDestination (cms::Session * session)` throw (cms::CMSEException) [virtual]
- 6.690.2.2** `ResolveProducerExecutor& activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::operator= (const ResolveProducerExecutor &)` [inline, protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.691 activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor`:

Public Member Functions

- `ResolveReceiveExecutor (CmsTemplate *parent, const std::string &selector, bool noLocal, const std::string &destinationName)`
- `virtual ~ResolveReceiveExecutor ()`
- `virtual cms::Destination * getDestination (cms::Session *session) throw (cms::CMSEException)`

Protected Member Functions

- `ResolveReceiveExecutor & operator= (const ResolveReceiveExecutor &)`

6.691.1 Constructor & Destructor Documentation

6.691.1.1 `activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::ResolveReceiveExecutor`
 (`CmsTemplate * parent`, `const std::string & selector`, `bool noLocal`,
`const std::string & destinationName`) `[inline]`

6.691.1.2 `virtual`
`activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::~~ResolveReceiveExecutor`
 () `[inline, virtual]`

6.691.2 Member Function Documentation

6.691.2.1 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::getDestination`
 (`cms::Session * session`) `throw (cms::CMSException) [virtual]`

6.691.2.2 `ResolveReceiveExecutor& activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::operator=` (`const ResolveReceiveExecutor &`) `[inline, protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.692 decaf::internal::util::Resource Class Reference

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

```
#include <src/main/decaf/internal/util/Resource.h>
```

Inheritance diagram for `decaf::internal::util::Resource`:

Public Member Functions

- `virtual ~Resource ()`

6.692.1 Detailed Description

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

Since

1.0

6.692.2 Constructor & Destructor Documentation

6.692.2.1 virtual decaf::internal::util::Resource::~~Resource () [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**Resource.h**

6.693 decaf::internal::util::ResourceLifecycleManager Class Reference

```
#include <src/main/decaf/internal/util/ResourceLifecycleManager.h>
```

Public Member Functions

- **ResourceLifecycleManager** ()
- virtual **~ResourceLifecycleManager** ()
- virtual void **addResource** (**Resource** *value)

Protected Member Functions

- virtual void **destroyResources** ()

6.693.1 Detailed Description

Since

1.0

6.693.2 Constructor & Destructor Documentation

6.693.2.1 decaf::internal::util::ResourceLifecycleManager::ResourceLifecycleManager ()

6.693.2.2 virtual decaf::internal::util::ResourceLifecycleManager::~~ResourceLifecycleManager () [virtual]

6.693.3 Member Function Documentation

6.693.3.1 virtual void decaf::internal::util::ResourceLifecycleManager::addResource (**Resource** * *value*) [virtual]

6.693.3.2 virtual void decaf::internal::util::ResourceLifecycleManager::destroyResources () [protected, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**ResourceLifecycleManager.h**

6.694 activemq::cmsutil::ResourceLifecycleManager Class Reference

Manages the lifecycle of a set of CMS resources.

```
#include <src/main/activemq/cmsutil/ResourceLifecycleManager.h>
```

Public Member Functions

- **ResourceLifecycleManager** ()
- virtual **~ResourceLifecycleManager** ()
*Destructor - calls **destroy***
- void **addConnection** (cms::Connection *connection) throw (cms::CMSEException)
Adds a connection so that its life will be managed by this object.
- void **addSession** (cms::Session *session) throw (cms::CMSEException)
Adds a session so that its life will be managed by this object.
- void **addDestination** (cms::Destination *dest) throw (cms::CMSEException)
Adds a destination so that its life will be managed by this object.
- void **addMessageProducer** (cms::MessageProducer *producer) throw (cms::CMSEException)
Adds a message producer so that its life will be managed by this object.
- void **addMessageConsumer** (cms::MessageConsumer *consumer) throw (cms::CMSEException)
Adds a message consumer so that its life will be managed by this object.
- void **destroy** () throw (cms::CMSEException)
Closes and destroys the contained CMS resources.
- void **releaseAll** ()
Releases all of the contained resources so that this object will no longer control their lifetimes.

Protected Member Functions

- **ResourceLifecycleManager** (const **ResourceLifecycleManager** &)
- **ResourceLifecycleManager** & **operator=** (const **ResourceLifecycleManager** &)

6.694.1 Detailed Description

Manages the lifecycle of a set of CMS resources. A call to **destroy** will close and destroy all of the contained resources in the appropriate manner.

6.694.2 Constructor & Destructor Documentation

6.694.2.1 `activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager (const ResourceLifecycleManager &) [inline, protected]`

6.694.2.2 `activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager ()`

6.694.2.3 `virtual
activemq::cmsutil::ResourceLifecycleManager::~~ResourceLifecycleManager () [virtual]`

Destructor - calls destroy

6.694.3 Member Function Documentation

6.694.3.1 `void activemq::cmsutil::ResourceLifecycleManager::addConnection (cms::Connection * connection) throw (cms::CMSException)`

Adds a connection so that its life will be managed by this object.

Parameters

connection the object to be managed

6.694.3.2 `void activemq::cmsutil::ResourceLifecycleManager::addDestination (cms::Destination * dest) throw (cms::CMSException)`

Adds a destination so that its life will be managed by this object.

Parameters

dest the object to be managed

6.694.3.3 `void activemq::cmsutil::ResourceLifecycleManager::addMessageConsumer (cms::MessageConsumer * consumer) throw (cms::CMSException)`

Adds a message consumer so that its life will be managed by this object.

Parameters

consumer the object to be managed

6.694.3.4 `void activemq::cmsutil::ResourceLifecycleManager::addMessageProducer (cms::MessageProducer * producer) throw (cms::CMSException)`

Adds a message producer so that its life will be managed by this object.

Parameters

producer the object to be managed

6.694.3.5 `void activemq::cmsutil::ResourceLifecycleManager::addSession (cms::Session * session) throw (cms::CMSException)`

Adds a session so that its life will be managed by this object.

Parameters

session the object to be managed

6.694.3.6 `void activemq::cmsutil::ResourceLifecycleManager::destroy () throw (cms::CMSException)`

Closes and destroys the contained CMS resources.

Exceptions

cms::CMSException (p. 1074) thrown if an error occurs.

6.694.3.7 `ResourceLifecycleManager& activemq::cmsutil::ResourceLifecycleManager::operator= (const ResourceLifecycleManager &) [inline, protected]`

6.694.3.8 `void activemq::cmsutil::ResourceLifecycleManager::releaseAll ()`

Releases all of the contained resources so that this object will no longer control their lifetimes.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/ResourceLifecycleManager.h`

6.695 activemq::commands::Response Class Reference

```
#include <src/main/activemq/commands/Response.h>
```

Inheritance diagram for `activemq::commands::Response`:

Public Member Functions

- `Response ()`
- `virtual ~Response ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual Response * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- `virtual void copyDataStructure (const DataStructure *src)`

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

- virtual int **getCorrelationId** () const

- virtual void **setCorrelationId** (int correlationId)

- virtual bool **isResponse** () const

- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_RESPONSE** = 30

Protected Attributes

- int correlationId

6.695.1 Constructor & Destructor Documentation

6.695.1.1 activemq::commands::Response::Response ()

6.695.1.2 virtual activemq::commands::Response::~~Response () [virtual]

6.695.2 Member Function Documentation

6.695.2.1 virtual Response* activemq::commands::Response::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1424), **activemq::commands::DataResponse** (p. 1478), **activemq::commands::ExceptionResponse** (p. 1721), and **activemq::commands::IntegerResponse** (p. 1957).

6.695.2.2 `virtual void activemq::commands::Response::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

Reimplemented in `activemq::commands::DataArrayResponse` (p. 1424), `activemq::commands::DataResponse` (p. 1478), `activemq::commands::ExceptionResponse` (p. 1721), and `activemq::commands::IntegerResponse` (p. 1957).

6.695.2.3 `virtual bool activemq::commands::Response::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

Reimplemented in `activemq::commands::DataArrayResponse` (p. 1424), `activemq::commands::DataResponse` (p. 1478), `activemq::commands::ExceptionResponse` (p. 1721), and `activemq::commands::IntegerResponse` (p. 1957).

6.695.2.4 `virtual int activemq::commands::Response::getCorrelationId () const [virtual]`

6.695.2.5 `virtual unsigned char activemq::commands::Response::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaller share.

Returns

new `DataStructure` (p. 1553) type copy.

Implements `activemq::commands::DataStructure` (p. 1557).

Reimplemented in `activemq::commands::DataArrayResponse` (p. 1425), `activemq::commands::DataResponse` (p. 1479), `activemq::commands::ExceptionResponse` (p. 1721), and `activemq::commands::IntegerResponse` (p. 1958).

6.695.2.6 `virtual bool activemq::commands::Response::isResponse () const`
`[inline, virtual]`

Returns

an answer of true to the `isResponse()` (p. 3078) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 699).

6.695.2.7 `virtual void activemq::commands::Response::setCorrelationId (int`
`correlationId) [virtual]`

6.695.2.8 `virtual std::string activemq::commands::Response::toString () const`
`[virtual]`

Returns a string containing the information for this `DataSet` (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 700).

Reimplemented in `activemq::commands::DataArrayResponse`
 (p. 1425), `activemq::commands::DataResponse` (p. 1479), `ac-`
`tivemq::commands::ExceptionResponse` (p. 1722), and `ac-`
`tivemq::commands::IntegerResponse` (p. 1958).

6.695.2.9 `virtual Pointer<Command> activemq::commands::Response::visit`
`(activemq::state::CommandVisitor * visitor) throw (`
`exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a `Response` (p. 3076) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1112).

6.695.3 Field Documentation

6.695.3.1 `int activemq::commands::Response::correlationId` `[protected]`

6.695.3.2 `const unsigned char activemq::commands::Response::ID_RESPONSE =`
`30 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/Response.h`

6.696 activemq::transport::mock::ResponseBuilder Class Reference

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

```
#include <src/main/activemq/transport/mock/ResponseBuilder.h>
```

Inheritance diagram for `activemq::transport::mock::ResponseBuilder`:

Public Member Functions

- virtual `~ResponseBuilder()`
- virtual `Pointer< Response > buildResponse (const Pointer< Command > &command)=0`
Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.
- virtual void `buildIncomingCommands (const Pointer< Command > &command, decaf::util::StlQueue< Pointer< Command > > &queue)=0`
*When called the **ResponseBuilder** (p. 3079) must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.*

6.696.1 Detailed Description

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

6.696.2 Constructor & Destructor Documentation

6.696.2.1 virtual `activemq::transport::mock::ResponseBuilder::~ResponseBuilder ()` [inline, virtual]

6.696.3 Member Function Documentation

6.696.3.1 virtual void `activemq::transport::mock::ResponseBuilder::buildIncomingCommands (const Pointer< Command > & command, decaf::util::StlQueue< Pointer< Command > > & queue)` [pure virtual]

When called the **ResponseBuilder** (p. 3079) must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

Parameters

- command* - The Command being sent to the Broker.
- queue* - Queue of Command sent back from the broker.

Implemented in `activemq::wireformat::openwire::OpenWireResponseBuilder` (p. 2718).

6.696.3.2 `virtual Pointer<Response> activemq::transport::mock::ResponseBuilder::buildResponse (const Pointer< Command > & command)` [pure virtual]

Given a `Command`, check if it requires a response and return the appropriate `Response` that the Broker would send for this `Command`.

Parameters

command - The command to build a response for

Returns

A `Response` object pointer, or `NULL` if no response.

Implemented in `activemq::wireformat::openwire::OpenWireResponseBuilder` (p. 2718).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/ResponseBuilder.h`

6.697 `activemq::transport::correlator::ResponseCorrelator` Class Reference

This type of transport filter is responsible for correlating asynchronous responses with requests.

`#include <src/main/activemq/transport/correlator/ResponseCorrelator.h>`

Inheritance diagram for `activemq::transport::correlator::ResponseCorrelator`:

Public Member Functions

- **`ResponseCorrelator (const Pointer< Transport > &next)`**
Constructor.
- `virtual ~ResponseCorrelator ()`
- `virtual void oneway (const Pointer< Command > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)`
Sends a one-way command.
- `virtual Pointer< Response > request (const Pointer< Command > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)`
Sends the given request to the server and waits for the response.
- `virtual Pointer< Response > request (const Pointer< Command > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)`
Sends the given request to the server and waits for the response.
- `virtual void onCommand (const Pointer< Command > &command)`

This is called in the context of the nested transport's reading thread.

- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.
- virtual void **onTransportException** (Transport *source, const decaf::lang::Exception &ex)
Event handler for an exception from a command transport.

6.697.1 Detailed Description

This type of transport filter is responsible for correlating asynchronous responses with requests. Non-response messages are simply sent directly to the CommandListener. It owns the transport that it

6.697.2 Constructor & Destructor Documentation

6.697.2.1 activemq::transport::correlator::ResponseCorrelator::ResponseCorrelator (const Pointer< Transport > & next)

Constructor.

Parameters

next the next transport in the chain

6.697.2.2 virtual activemq::transport::correlator::ResponseCorrelator::~ResponseCorrelator () [virtual]

6.697.3 Member Function Documentation

6.697.3.1 virtual void activemq::transport::correlator::ResponseCorrelator::close () throw (decaf::io::IOException) [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if errors occur.

Reimplemented from **activemq::transport::TransportFilter** (p. 3638).

6.697.3.2 virtual void activemq::transport::correlator::ResponseCorrelator::onCommand (const Pointer< Command > & *command*) [virtual]

This is called in the context of the nested transport's reading thread.

In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

Parameters

command the received from the nested transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3640).

6.697.3.3 virtual void activemq::transport::correlator::ResponseCorrelator::oneway (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3640).

6.697.3.4 virtual void activemq::transport::correlator::ResponseCorrelator::onTransportException (Transport * *source*, const decaf::lang::Exception & *ex*) [virtual]

Event handler for an exception from a command transport.

Parameters

source The source of the exception

ex The exception.

6.697.3.5 virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given request to the server and waits for the response.

Parameters

command The request to send.

Returns

the response from the server.

Exceptions

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3642).

```
6.697.3.6 virtual Pointer<Response> ac-
    tivemq::transport::correlator::ResponseCorrelator::request
    ( const Pointer< Command > & command, un-
      signed int timeout ) throw ( decaf::io::IOException,
      decaf::lang::exceptions::UnsupportedOperationException ) [virtual]
```

Sends the given request to the server and waits for the response.

Parameters

command The request to send.

timeout The time to wait for a response.

Returns

the response from the server.

Exceptions

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3641).

```
6.697.3.7 virtual void activemq::transport::correlator::ResponseCorrelator::start (
    ) throw ( decaf::io::IOException ) [virtual]
```

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions

IOException if an error occurs or if this transport has already been closed.

Reimplemented from **activemq::transport::TransportFilter** (p. 3642).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/correlator/**ResponseCorrelator.h**

6.698 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3085).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.698.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3085). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.698.2 Constructor & Destructor Documentation

6.698.2.1 `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::ResponseMarshaller () [inline]`

6.698.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::~~ResponseMarshaller () [inline, virtual]`

6.698.3 Member Function Documentation

6.698.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1435), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1497), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1740), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1972).

6.698.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1435), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1497), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1740), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1972).

6.698.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1435), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1497), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1740), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1972).

```
6.698.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1436), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1498), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1741), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1973).

```
6.698.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1436), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1498), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1741), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1973).

6.698.3.6 virtual void `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal2`
 (`OpenWireFormat * wireFormat`, `commands::DataStructure`
 * `dataStructure`, `decaf::io::DataOutputStream * dataOut`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1437), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1499), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1742), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1974).

6.698.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1437), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1499), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1742), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 1974).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ResponseMarshaller.h**

6.699 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3089).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.699.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3089). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.699.2 Constructor & Destructor Documentation

6.699.2.1 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::ResponseMarshaller () [inline]

6.699.2.2 virtual
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::~~ResponseMarshaller () [inline, virtual]

6.699.3 Member Function Documentation

6.699.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1427), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1489), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1728), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1964).

6.699.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1427), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1489), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1728), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1964).

6.699.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 736).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1427), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1489), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1728), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1964).

6.699.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1428), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1490), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1729), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1965).

6.699.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1428), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1490), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1729), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1965).

```

6.699.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1429), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1491), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1730), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1966).

```

6.699.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 740).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1429), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1491), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller`

(p. 1730), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1966).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h`

6.700 `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `ResponseMarshaller` (p. 3093).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller`:

Public Member Functions

- `ResponseMarshaller ()`
- `virtual ~ResponseMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.700.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3093). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.700.2 Constructor & Destructor Documentation

6.700.2.1 `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::ResponseMarshaller`
() [inline]

6.700.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::~~ResponseMarshaller`
() [inline, virtual]

6.700.3 Member Function Documentation

6.700.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::createObject`
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1443), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1481), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1736), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1980).

6.700.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::getDataStructureType`
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1443), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1481), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller`

(p. 1736), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1980).

6.700.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1443), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1481), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1736), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1980).

6.700.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1444), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1482), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1737), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1981).

```

6.700.3.5  virtual int ac-
            tivemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, utils::BooleanStream * bs ) throw (
            decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1444), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1482), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1737), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1981).

```

6.700.3.6  virtual void ac-
            tivemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1445), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller`

(p. 1483), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1738), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1982).

6.700.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 727).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1445), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1483), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1738), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1982).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h`

6.701 `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `ResponseMarshaller` (p. 3098).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller`:

Public Member Functions

- `ResponseMarshaller ()`
- `virtual ~ResponseMarshaller ()`
- `virtual commands::DataStructure * createObject () const`

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.701.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3098). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.701.2 Constructor & Destructor Documentation

6.701.2.1 `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::ResponseMarshaller () [inline]`

6.701.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::~~ResponseMarshaller () [inline, virtual]`

6.701.3 Member Function Documentation

6.701.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1431), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1493), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1732), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1968).

6.701.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1431), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1493), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1732), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1968).

6.701.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1431), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1493), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1732), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1968).

```
6.701.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1432), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1494), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1733), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1969).

```
6.701.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1432), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1494), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1733), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1969).

6.701.3.6 virtual void `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal2`
 (`OpenWireFormat * wireFormat`, `commands::DataStructure`
 * `dataStructure`, `decaf::io::DataOutputStream * dataOut`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1433), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1495), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1734), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1970).


```
6.701.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1433), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1495), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1734), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1970).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h`

6.702 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3102).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.702.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3102). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.702.2 Constructor & Destructor Documentation

6.702.2.1 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::ResponseMarshaller () [inline]

6.702.2.2 virtual
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::~~ResponseMarshaller () [inline, virtual]

6.702.3 Member Function Documentation

6.702.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1439), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1501), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1744), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1976).

6.702.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1439), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1501), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1744), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1976).

6.702.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 716).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1439), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1501), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1744), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1976).

6.702.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1440), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1745), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1977).

6.702.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1440), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1745), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1977).

```

6.702.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1441), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1503), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1746), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1978).

```

6.702.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 720).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1441), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1503), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller`

(p. 1746), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1978).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h`

6.703 `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `ResponseMarshaller` (p. 3107).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller`:

Public Member Functions

- `ResponseMarshaller ()`
- `virtual ~ResponseMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.703.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3107). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.703.2 Constructor & Destructor Documentation

6.703.2.1 `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::ResponseMarshaller () [inline]`

6.703.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::~~ResponseMarshaller () [inline, virtual]`

6.703.3 Member Function Documentation

6.703.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1447), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1485), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1724), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1960).

6.703.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1447), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1485), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller`

(p. 1724), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1960).

6.703.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1447), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1485), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1724), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1960).

6.703.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1448), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1486), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1725), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1961).


```

6.703.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1448), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1486), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1725), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1961).

```

6.703.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1449), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller`

(p. 1487), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1726), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1962).

6.703.3.7 `virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 733).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1449), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1487), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1726), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 1962).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h`

6.704 decaf::lang::Runnable Class Reference

Interface for a runnable object - defines a task that can be run by a thread.

```
#include <src/main/decaf/lang/Runnable.h>
```

Inheritance diagram for `decaf::lang::Runnable`:

Public Member Functions

- `virtual ~Runnable ()`
- `virtual void run ()=0`

Run method - called by the *Thread* (p. 3520) class in the context of the thread.

6.704.1 Detailed Description

Interface for a runnable object - defines a task that can be run by a thread.

6.704.2 Constructor & Destructor Documentation

6.704.2.1 virtual decaf::lang::Runnable::~~Runnable () [inline, virtual]

6.704.3 Member Function Documentation

6.704.3.1 virtual void decaf::lang::Runnable::run () [pure virtual]

Run method - called by the **Thread** (p. 3520) class in the context of the thread.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 1134), **activemq::threads::DedicatedTaskRunner** (p. 1565), **activemq::transport::inactivity::ReadChecker** (p. 2960), **activemq::transport::inactivity::WriteChecker** (p. 3755), **activemq::transport::IOTransport** (p. 2010), **activemq::transport::mock::InternalCommandListener** (p. 1987), **decaf::lang::Thread** (p. 3527), and **decaf::util::concurrent::PooledThread** (p. 2778).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Runnable.h**

6.705 decaf::lang::Runtime Class Reference

```
#include <src/main/decaf/lang/Runtime.h>
```

Inheritance diagram for decaf::lang::Runtime:

Public Member Functions

- virtual ~**Runtime** ()

Static Public Member Functions

- static **Runtime** * **getRuntime** ()
*Gets the single instance of the Decaf **Runtime** (p. 3112) for this Process.*
- static void **initializeRuntime** (int argc, char **argv)
Initialize the Decaf Library passing it the args that were passed to the application at startup.
- static void **initializeRuntime** ()
Initialize the Decaf Library.
- static void **shutdownRuntime** ()

Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

6.705.1 Constructor & Destructor Documentation

6.705.1.1 `virtual decaf::lang::Runtime::~~Runtime () [inline, virtual]`

6.705.2 Member Function Documentation

6.705.2.1 `static Runtime* decaf::lang::Runtime::getRuntime () [static]`

Gets the single instance of the Decaf **Runtime** (p. 3112) for this Process.

Returns

pointer to the single Decaf **Runtime** (p. 3112) instance that exists for this process

6.705.2.2 `static void decaf::lang::Runtime::initializeRuntime (int argc, char ** argv) [static]`

Initialize the Decaf Library passing it the args that were passed to the application at startup.

Parameters

argc - The number of args passed

argv - Array of char* values passed to the Process on start.

Exceptions

runtime_error if the library is already initialized or an error occurs during initialization.

6.705.2.3 `static void decaf::lang::Runtime::initializeRuntime () [static]`

Initialize the Decaf Library.

Exceptions

runtime_error if the library is already initialized or an error occurs during initialization.

6.705.2.4 `static void decaf::lang::Runtime::shutdownRuntime () [static]`

Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

Exceptions

runtime_error if the library has not already been initialized or an error occurs during shutdown.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Runtime.h`

6.706 decaf::lang::exceptions::RuntimeException Class Reference

#include <src/main/decaf/lang/exceptions/RuntimeException.h>

Inheritance diagram for decaf::lang::exceptions::RuntimeException:

Public Member Functions

- **RuntimeException** () throw ()
Default Constructor.
- **RuntimeException** (const **Exception** &ex) throw ()
Conversion Constructor from some other ActiveMQException.
- **RuntimeException** (const **RuntimeException** &ex) throw ()
Copy Constructor.
- **RuntimeException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **RuntimeException** (const std::exception *cause) throw ()
Constructor.
- **RuntimeException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **RuntimeException** * **clone** () const
Clones this exception.
- virtual ~**RuntimeException** () throw ()

6.706.1 Constructor & Destructor Documentation

6.706.1.1 decaf::lang::exceptions::RuntimeException::RuntimeException () throw () [inline]

Default Constructor.

6.706.1.2 decaf::lang::exceptions::RuntimeException::RuntimeException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other ActiveMQException.

Parameters

ex The **Exception** (p.1712) whose data is to be copied into this one.

6.706.1.3 `decaf::lang::exceptions::RuntimeException::RuntimeException (const RuntimeException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex The **Exception** (p. 1712) whose data is to be copied into this one.

6.706.1.4 `decaf::lang::exceptions::RuntimeException::RuntimeException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.706.1.5 `decaf::lang::exceptions::RuntimeException::RuntimeException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause **Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.706.1.6 `decaf::lang::exceptions::RuntimeException::RuntimeException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.706.1.7 virtual decaf::lang::exceptions::RuntimeException::~~RuntimeException () throw () [inline, virtual]

6.706.2 Member Function Documentation

6.706.2.1 virtual RuntimeException* decaf::lang::exceptions::RuntimeException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p.1712) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1715).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**RuntimeException.h**

6.707 decaf::security::SecureRandom Class Reference

```
#include <src/main/decaf/security/SecureRandom.h>
```

Inheritance diagram for decaf::security::SecureRandom:

Public Member Functions

- **SecureRandom** ()
Creates a new instance of a secure random number generator that implements the default random number algorithm.
- **SecureRandom** (const std::vector< unsigned char > &seed)
Creates a new instance of a secure random number generator that implements the default random number algorithm.
- **SecureRandom** (const unsigned char *seed, int size)
Creates a new instance of a secure random number generator that implements the default random number algorithm.
- virtual ~**SecureRandom** ()
- virtual void **nextBytes** (std::vector< unsigned char > &buf)
Modifies the byte array by a random sequence of bytes generated by this random number generator.
Parameters
buf non-null array to contain the new random bytes

See also

next (p. 2954)

- virtual void **nextBytes** (unsigned char *buf, int size)

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

buf non-null array to contain the new random bytes

See also

next (p. 2954)

Exceptions

NullPointerException if *buff* is *NULL*

IllegalArgumentException if *size* is negative

- virtual void **setSeed** (unsigned long long seed)

Modifies the seed using linear congruential formula presented in The Art of Computer Programming, Volume 2, Section 3.2.1.

Parameters

seed the seed that alters the state of the random number generator

See also

next (p. 2954)

Random() (p. 2954)

Random(long)

- virtual void **setSeed** (const std::vector< unsigned char > &seed)

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

- virtual void **setSeed** (const unsigned char *seed, int size)

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

Protected Member Functions

- virtual int **next** (int bits)

*Answers a pseudo-random uniformly distributed **int** value of the number of bits specified by the argument **bits** as described by Donald E.*

Knuth in The Art of Computer Programming, Volume 2: Seminumerical Algorithms, section 3.2.1.

Returns

int a pseudo-random generated int number

Parameters

bits number of bits of the returned value

See also

nextBytes (p. 2955)

nextDouble (p. 2956)

nextFloat (p. 2956)

nextInt() (p. 2956)

nextInt(int) (p. 2957)

nextGaussian (p. 2956)

nextLong (p. 2957)

6.707.1 Detailed Description

Since

1.0

6.707.2 Constructor & Destructor Documentation

6.707.2.1 decaf::security::SecureRandom::SecureRandom ()

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 3116) instance that is created with this constructor is unseeded and can be seeded by calling the `setSeed` method. Calls to `nextBytes` on an unseeded **SecureRandom** (p. 3116) result in the object seeding itself.

6.707.2.2 decaf::security::SecureRandom::SecureRandom (const std::vector< unsigned char > & seed)

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 3116) instance created by this constructor is seeded using the passed byte array.

Parameters

seed The seed bytes to use to seed this secure random number generator.

6.707.2.3 decaf::security::SecureRandom::SecureRandom (const unsigned char * seed, int size)

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 3116) instance created by this constructor is seeded using the passed byte array.

Parameters

seed The seed bytes to use to seed this secure random number generator.

size The number of bytes in the seed buffer.

Exceptions

NullPointerException if the seed buffer is NULL.

IllegalArgumentException if the size value is negative.

6.707.2.4 `virtual decaf::security::SecureRandom::~~SecureRandom ()` [virtual]

6.707.3 Member Function Documentation

6.707.3.1 `virtual int decaf::security::SecureRandom::next (int bits)` [protected, virtual]

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E.

Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

Returns

int a pseudo-random generated int number

Parameters

bits number of bits of the returned value

See also

`nextBytes` (p. 2955)
`nextDouble` (p. 2956)
`nextFloat` (p. 2956)
`nextInt()` (p. 2956)
`nextInt(int)` (p. 2957)
`nextGaussian` (p. 2956)
`nextLong` (p. 2957)

Reimplemented from `decaf::util::Random` (p. 2954).

6.707.3.2 `virtual void decaf::security::SecureRandom::nextBytes (unsigned char * buf, int size)` [virtual]

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

buf non-null array to contain the new random bytes

See also

`next` (p. 2954)

Exceptions

NullPointerException if *buf* is NULL
IllegalArgumentException if *size* is negative

Reimplemented from `decaf::util::Random` (p. 2955).

6.707.3.3 `virtual void decaf::security::SecureRandom::nextBytes (std::vector< unsigned char > & buf)` [virtual]

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

buf non-null array to contain the new random bytes

See also

`next` (p. 2954)

Reimplemented from `decaf::util::Random` (p. 2955).

6.707.3.4 virtual void decaf::security::SecureRandom::setSeed (const std::vector< unsigned char > & *seed*) [virtual]

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

Parameters

seed A vector of bytes that is used update the seed of the RNG.

6.707.3.5 virtual void decaf::security::SecureRandom::setSeed (const unsigned char * *seed*, int *size*) [virtual]

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

Parameters

seed The seed bytes to use to seed this secure random number generator.

size The number of bytes in the seed buffer.

Exceptions

NullPointerException if the seed buffer is NULL.

IllegalArgumentException if the size value is negative.

6.707.3.6 virtual void decaf::security::SecureRandom::setSeed (unsigned long long *seed*) [virtual]

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

Parameters

seed the seed that alters the state of the random number generator

See also

`next` (p. 2954)

`Random()` (p. 2954)

`Random(long)`

Reimplemented from `decaf::util::Random` (p. 2957).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/SecureRandom.h`

6.708 decaf::internal::security::SecureRandomImpl Class Reference

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

```
#include <src/main/decaf/internal/security/unix/SecureRandomImpl.h>
```

Inheritance diagram for decaf::internal::security::SecureRandomImpl:

Public Member Functions

- **SecureRandomImpl** ()
- virtual **~SecureRandomImpl** ()
- virtual void **providerSetSeed** (const unsigned char *seed, int size)

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.
- virtual void **providerNextBytes** (unsigned char *bytes, int numBytes)

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.
- virtual unsigned char * **providerGenerateSeed** (int numBytes)

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.
- **SecureRandomImpl** ()
- virtual **~SecureRandomImpl** ()
- virtual void **providerSetSeed** (const unsigned char *seed, int size)

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.
- virtual void **providerNextBytes** (unsigned char *bytes, int numBytes)

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.
- virtual unsigned char * **providerGenerateSeed** (int numBytes)

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

6.708.1 Detailed Description

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources. Secure Random Number Generator for Windows based platforms that attempts to obtain secure bytes with high entropy from known sources.

If the platform does not have a source of secure bytes then the platform random number generator is used if one exists otherwise the Decaf RNG is used as a last resort.

Since

1.0

6.708.2 Constructor & Destructor Documentation

6.708.2.1 decaf::internal::security::SecureRandomImpl::SecureRandomImpl ()

6.708.2.2 virtual
decaf::internal::security::SecureRandomImpl::~SecureRandomImpl ()
[virtual]

6.708.2.3 decaf::internal::security::SecureRandomImpl::SecureRandomImpl ()

6.708.2.4 virtual
decaf::internal::security::SecureRandomImpl::~SecureRandomImpl ()
[virtual]

6.708.3 Member Function Documentation

6.708.3.1 virtual unsigned char* decaf::internal::security::SecureRandomImpl::providerGenerateSeed (int *numBytes*) [virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

Parameters

numBytes The number of bytes that should be generated for the new seed array.

Implements decaf::security::SecureRandomSpi (p. 3125).

6.708.3.2 virtual unsigned char* decaf::internal::security::SecureRandomImpl::providerGenerateSeed (int *numBytes*) [virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

Parameters

numBytes The number of bytes that should be generated for the new seed array.

Implements decaf::security::SecureRandomSpi (p. 3125).

6.708.3.3 `virtual void decaf::internal::security::SecureRandomImpl::providerNextBytes (unsigned char * bytes, int numBytes) [virtual]`

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

Parameters

bytes The array that will be filled with random bytes equal to size.

numBytes The number of bytes to generate and write into the bytes array.

Implements `decaf::security::SecureRandomSpi` (p. 3125).

6.708.3.4 `virtual void decaf::internal::security::SecureRandomImpl::providerNextBytes (unsigned char * bytes, int numBytes) [virtual]`

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

Parameters

bytes The array that will be filled with random bytes equal to size.

numBytes The number of bytes to generate and write into the bytes array.

Implements `decaf::security::SecureRandomSpi` (p. 3125).

6.708.3.5 `virtual void decaf::internal::security::SecureRandomImpl::providerSetSeed (const unsigned char * seed, int size) [virtual]`

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

Parameters

seed The array of bytes used to update the generators seed.

size The size of the passed byte array.

Implements `decaf::security::SecureRandomSpi` (p. 3126).

6.708.3.6 `virtual void decaf::internal::security::SecureRandomImpl::providerSetSeed (const unsigned char * seed, int size) [virtual]`

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

Parameters

seed The array of bytes used to update the generators seed.

size The size of the passed byte array.

Implements **decaf::security::SecureRandomSpi** (p. 3126).

The documentation for this class was generated from the following files:

- `src/main/decaf/internal/security/unix/SecureRandomImpl.h`
- `src/main/decaf/internal/security/windows/SecureRandomImpl.h`

6.709 decaf::security::SecureRandomSpi Class Reference

Interface class used by Security Service Providers to implement a source of secure random bytes.

`#include <src/main/decaf/security/SecureRandomSpi.h>`

Inheritance diagram for `decaf::security::SecureRandomSpi`:

Public Member Functions

- **SecureRandomSpi** ()
- virtual **~SecureRandomSpi** ()
- virtual void **providerSetSeed** (const unsigned char *seed, int size)=0

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

- virtual void **providerNextBytes** (unsigned char *bytes, int numBytes)=0

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

- virtual unsigned char * **providerGenerateSeed** (int numBytes)=0

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

6.709.1 Detailed Description

Interface class used by Security Service Providers to implement a source of secure random bytes.

Since

1.0

6.709.2 Constructor & Destructor Documentation

6.709.2.1 `decaf::security::SecureRandomSpi::SecureRandomSpi ()`

6.709.2.2 `virtual decaf::security::SecureRandomSpi::~~SecureRandomSpi ()`
[virtual]

6.709.3 Member Function Documentation

6.709.3.1 `virtual unsigned char* decaf::security::SecureRandomSpi::providerGenerateSeed (int numBytes)` [pure virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

Parameters

numBytes The number of bytes that should be generated for the new seed array.

Implemented in `decaf::internal::security::SecureRandomImpl` (p. 3123), and `decaf::internal::security::SecureRandomImpl` (p. 3123).

6.709.3.2 `virtual void decaf::security::SecureRandomSpi::providerNextBytes (unsigned char * bytes, int numBytes)` [pure virtual]

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

Parameters

bytes The array that will be filled with random bytes equal to size.

numBytes The number of bytes to generate and write into the bytes array.

Implemented in `decaf::internal::security::SecureRandomImpl` (p. 3123), and `decaf::internal::security::SecureRandomImpl` (p. 3123).

6.709.3.3 `virtual void decaf::security::SecureRandomSpi::providerSetSeed (const unsigned char * seed, int size)` [pure virtual]

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

Parameters

seed The array of bytes used to update the generators seed.

size The size of the passed byte array.

Implemented in `decaf::internal::security::SecureRandomImpl` (p. 3124), and `decaf::internal::security::SecureRandomImpl` (p. 3124).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/SecureRandomSpi.h`

6.710 decaf::util::concurrent::Semaphore Class Reference

A counting semaphore.

```
#include <src/main/decaf/util/concurrent/Semaphore.h>
```

Public Member Functions

- **Semaphore** (int permits)
*Creates a **Semaphore** (p. 3126) with the given number of permits and nonfair fairness setting.*
- **Semaphore** (int permits, bool fair)
*Creates a **Semaphore** (p. 3126) with the given number of permits and the given fairness setting.*
- virtual **~Semaphore** ()
- void **acquire** () throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.
- void **acquireUninterruptibly** () throw (decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, blocking until one is available.
- bool **tryAcquire** () throw (decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, only if one is available at the time of invocation.
- bool **tryAcquire** (long long timeout, const **TimeUnit** &unit) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.
- void **release** () throw (decaf::lang::exceptions::RuntimeException)
Releases a permit, returning it to the semaphore.
- void **acquire** (int permits) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.
- void **acquireUninterruptibly** (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Acquires the given number of permits from this semaphore, blocking until all are available.

- **bool tryAcquire** (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.
- **bool tryAcquire** (int permits, long long timeout, const **TimeUnit** &unit) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.
- **void release** (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Releases the given number of permits, returning them to the semaphore.
- **int availablePermits** () const
Returns the current number of permits available in this semaphore.
- **int drainPermits** () throw (decaf::lang::exceptions::RuntimeException)
Acquires and returns all permits that are immediately available.
- **bool isFair** () const
- **std::string toString** () const
Returns a string identifying this semaphore, as well as its state.

6.710.1 Detailed Description

A counting semaphore. Conceptually, a semaphore maintains a set of permits. Each **acquire()** (p. 3129) blocks if necessary until a permit is available, and then takes it. Each **release()** (p. 3132) adds a permit, potentially releasing a blocking acquirer. However, no actual permit objects are used; the **Semaphore** (p. 3126) just keeps a count of the number available and acts accordingly.

Semaphores are often used to restrict the number of threads than can access some (physical or logical) resource.

```
class Pool { private:
static const int MAX_AVAILABLE = 100; Semaphore (p. 3126) available;
std::vector<std::string> items; std::vector<bool> used;
Mutex (p. 2604) lock;
public:
Pool() : available( MAX_AVAILABLE, true ) { used.resize( MAX_AVAILABLE ); items.resize(
MAX_AVAILABLE ); }
std::string getItem() throws InterruptedException { available.acquire(); return getNextAvailableItem(); }
void putItem( std::string x ) { if( markAsUnused(x) ) { available.release(); } }
std::string getNextAvailableItem() {
```

```
synchronized( &lock ) (p. 4287) { for( int i = 0; i < MAX_AVAILABLE; ++i ) { if( !used[i] )
{ used[i] = true; return items[i]; } }
```

```
return std::string(); // not reached }
```

```
bool markAsUnused( const std::string& item ) { synchronized( &lock ) (p. 4287) { for( int i =
0; i < MAX_AVAILABLE; ++i ) { if( item == items[i] ) { if( used[i] ) { used[i] = false; return
true; } else return false; } } } return false; } };
```

Before obtaining an item each thread must acquire a permit from the semaphore, guaranteeing that an item is available for use. When the thread has finished with the item it is returned back to the pool and a permit is returned to the semaphore, allowing another thread to acquire that item. Note that no synchronization lock is held when **acquire()** (p. 3129) is called as that would prevent an item from being returned to the pool. The semaphore encapsulates the synchronization needed to restrict access to the pool, separately from any synchronization needed to maintain the consistency of the pool itself.

A semaphore initialized to one, and which is used such that it only has at most one permit available, can serve as a mutual exclusion lock. This is more commonly known as a binary semaphore, because it only has two states: one permit available, or zero permits available. When used in this way, the binary semaphore has the property (unlike many **Lock** (p. 2228) implementations), that the "lock" can be released by a thread other than the owner (as semaphores have no notion of ownership). This can be useful in some specialized contexts, such as deadlock recovery.

The constructor for this class optionally accepts a fairness parameter. When set false, this class makes no guarantees about the order in which threads acquire permits. In particular, barging is permitted, that is, a thread invoking **acquire()** (p. 3129) can be allocated a permit ahead of a thread that has been waiting - logically the new thread places itself at the head of the queue of waiting threads. When fairness is set true, the semaphore guarantees that threads invoking any of the acquire methods are selected to obtain permits in the order in which their invocation of those methods was processed (first-in-first-out; FIFO). Note that FIFO ordering necessarily applies to specific internal points of execution within these methods. So, it is possible for one thread to invoke acquire before another, but reach the ordering point after the other, and similarly upon return from the method. Also note that the untimed tryAcquire methods do not honor the fairness setting, but will take any permits that are available.

Generally, semaphores used to control resource access should be initialized as fair, to ensure that no thread is starved out from accessing a resource. When using semaphores for other kinds of synchronization control, the throughput advantages of non-fair ordering often outweigh fairness considerations.

This class also provides convenience methods to acquire and release multiple permits at a time. Beware of the increased risk of indefinite postponement when these methods are used without fairness set true.

Since

1.0

6.710.2 Constructor & Destructor Documentation

6.710.2.1 decaf::util::concurrent::Semaphore::Semaphore (int *permits*)

Creates a **Semaphore** (p. 3126) with the given number of permits and nonfair fairness setting.

Parameters

permits the initial number of permits available. This value may be negative, in which case

releases must occur before any acquires will be granted.

6.710.2.2 `decaf::util::concurrent::Semaphore::Semaphore (int permits, bool fair)`

Creates a **Semaphore** (p. 3126) with the given number of permits and the given fairness setting.

Parameters

permits the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.

fair true if this semaphore will guarantee first-in first-out granting of permits under contention, else false

6.710.2.3 `virtual decaf::util::concurrent::Semaphore::~~Semaphore () [virtual]`

6.710.3 Member Function Documentation

6.710.3.1 `void decaf::util::concurrent::Semaphore::acquire () throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException)`

Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.

Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* Some other thread invokes the **release()** (p. 3132) method for this semaphore and the current thread is next to be assigned a permit; or * Some other thread interrupts the current thread.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting for a permit,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

Exceptions

InterruptedException - if the current thread is interrupted.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 3126).

6.710.3.2 `void decaf::util::concurrent::Semaphore::acquire (int permits) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)`

Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.

Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or * Some other thread interrupts the current thread.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting for a permit,

then InterruptedException is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread are instead assigned to other threads trying to acquire permits, as if permits had been made available by a call to **release()** (p. 3132).

Parameters

permits the number of permits to acquire.

Exceptions

InterruptedException if the current thread is interrupted.

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 3126).

6.710.3.3 void decaf::util::concurrent::Semaphore::acquireUninterruptibly () throw (decaf::lang::exceptions::RuntimeException)

Acquires a permit from this semaphore, blocking until one is available.

Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes the **release()** (p. 3132) method for this semaphore and the current thread is next to be assigned a permit.

If the current thread is interrupted while waiting for a permit then it will continue to wait, but the time at which the thread is assigned a permit may change compared to the time it would have received the permit had no interruption occurred. When the thread does return from this method its interrupt status will be set.

Exceptions

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 3126).

6.710.3.4 void decaf::util::concurrent::Semaphore::acquireUninterruptibly (int *permits*) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)

Acquires the given number of permits from this semaphore, blocking until all are available.

Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request.

If the current thread is interrupted while waiting for permits then it will continue to wait and its position in the queue is not affected. When the thread does return from this method its interrupt status will be set.

Parameters

permits the number of permits to acquire.

Exceptions

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 3126).

6.710.3.5 int decaf::util::concurrent::Semaphore::availablePermits () const

Returns the current number of permits available in this semaphore.

This method is typically used for debugging and testing purposes.

Returns

the number of permits available in this semaphore

6.710.3.6 int decaf::util::concurrent::Semaphore::drainPermits () throw (decaf::lang::exceptions::RuntimeException)

Acquires and returns all permits that are immediately available.

Returns

the number of permits acquired

Exceptions

RuntimeException if an unexpected error occurs while draining the **Semaphore** (p. 3126).

6.710.3.7 bool decaf::util::concurrent::Semaphore::isFair () const

Returns

true if this semaphore has fairness set true

6.710.3.8 void decaf::util::concurrent::Semaphore::release () throw (decaf::lang::exceptions::RuntimeException)

Releases a permit, returning it to the semaphore.

Releases a permit, increasing the number of available permits by one. If any threads are trying to acquire a permit, then one is selected and given the permit that was just released. That thread is (re)enabled for thread scheduling purposes.

There is no requirement that a thread that releases a permit must have acquired that permit by calling **acquire()** (p. 3129). Correct usage of a semaphore is established by programming convention in the application.

Exceptions

RuntimeException if an unexpected error occurs while releasing the **Semaphore** (p. 3126).

6.710.3.9 void decaf::util::concurrent::Semaphore::release (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)

Releases the given number of permits, returning them to the semaphore.

Releases the given number of permits, increasing the number of available permits by that amount. If any threads are trying to acquire permits, then one is selected and given the permits that were just released. If the number of available permits satisfies that thread's request then that thread is (re)enabled for thread scheduling purposes; otherwise the thread will wait until sufficient permits are available. If there are still permits available after this thread's request has been satisfied, then those permits are assigned in turn to other threads trying to acquire permits.

Parameters

permits the number of permits to release

Exceptions

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while releasing the **Semaphore** (p. 3126).

6.710.3.10 std::string decaf::util::concurrent::Semaphore::toString () const

Returns a string identifying this semaphore, as well as its state.

The state, in brackets, includes the String "Permits =" followed by the number of permits.

Returns

a string identifying this semaphore, as well as its state

6.710.3.11 `bool decaf::util::concurrent::Semaphore::tryAcquire (int
permits, long long timeout, const TimeUnit & unit)
throw (decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::RuntimeException)`

Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.

Acquires the given number of permits, if they are available and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or * Some other thread interrupts the current thread; or * The specified waiting time elapses.

If the permits are acquired then the value true is returned.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting to acquire the permits,

then InterruptedException is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 3132).

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 3132).

Parameters

permits the number of permits to acquire

timeout the maximum amount of time to wait to acquire the permits.

unit the units that the timeout param represents.

Returns

true if all permits were acquired and false if the waiting time elapsed before all permits were acquired

Exceptions

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 3126).

6.710.3.12 `bool decaf::util::concurrent::Semaphore::tryAcquire
(long long timeout, const TimeUnit & unit)
throw (decaf::lang::exceptions::InterruptedException,
decaf::lang::exceptions::RuntimeException)`

Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.

Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- * Some other thread invokes the **release()** (p. 3132) method for this semaphore and the current thread is next to be assigned a permit; or
- * Some other thread interrupts the current thread; or
- * The specified waiting time elapses.

If a permit is acquired then the value true is returned.

If the current thread:

- * has its interrupted status set on entry to this method; or
- * is interrupted while waiting to acquire a permit,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Parameters

timeout the maximum time to wait for a permit

unit the time unit of the timeout argument

Returns

true if a permit was acquired and false if the waiting time elapsed before a permit was acquired

Exceptions

InterruptedException if the current thread is interrupted.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 3126).

6.710.3.13 bool decaf::util::concurrent::Semaphore::tryAcquire () throw (decaf::lang::exceptions::RuntimeException)

Acquires a permit from this semaphore, only if one is available at the time of invocation.

Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then this method will return immediately with the value false.

Even when this semaphore has been set to use a fair ordering policy, a call to **tryAcquire()** (p. 3134) will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use **tryAcquire(0, TimeUnit.SECONDS)** (p. 3568) which is almost equivalent (it also detects interruption).

Returns

true if a permit was acquired and false otherwise

Exceptions

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 3126).

6.710.3.14 `bool decaf::util::concurrent::Semaphore::tryAcquire (int permits
) throw (decaf::lang::exceptions::IllegalArgumentException,
 decaf::lang::exceptions::RuntimeException)`

Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.

Acquires the given number of permits, if they are available, and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then this method will return immediately with the value false and the number of available permits is unchanged.

Even when this semaphore has been set to use a fair ordering policy, a call to tryAcquire will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use tryAcquire(permits, 0, **TimeUnit.SECONDS** (p. 3568)) which is almost equivalent (it also detects interruption).

Parameters

permits the number of permits to acquire

Returns

true if the permits were acquired and false otherwise.

Exceptions

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 3126).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Semaphore.h`

6.711 activemq::cmsutil::CmsTemplate::SendExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::SendExecutor:

Public Member Functions

- **SendExecutor** (**MessageCreator** *messageCreator, **CmsTemplate** *parent)
- virtual **~SendExecutor** ()
- virtual void **doInCms** (**cms::Session** *session, **cms::MessageProducer** *producer) throw (**cms::CMSException**)

Execute an action given a session and producer.

Protected Member Functions

- `SendExecutor` (`const SendExecutor &`)
- `SendExecutor & operator=` (`const SendExecutor &`)

6.711.1 Constructor & Destructor Documentation

- 6.711.1.1 `activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor (const SendExecutor &)` [inline, protected]
- 6.711.1.2 `activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor (MessageCreator * messageCreator, CmsTemplate * parent)` [inline]
- 6.711.1.3 `virtual activemq::cmsutil::CmsTemplate::SendExecutor::~~SendExecutor ()` [inline, virtual]

6.711.2 Member Function Documentation

- 6.711.2.1 `virtual void activemq::cmsutil::CmsTemplate::SendExecutor::doInCms (cms::Session * session, cms::MessageProducer * producer) throw (cms::CMSException)` [inline, virtual]

Execute an action given a session and producer.

Parameters

- session* the CMS Session
producer the CMS Producer

Exceptions

- cms::CMSException* (p. 1074) if thrown by CMS API methods

Implements `activemq::cmsutil::ProducerCallback` (p. 2867).

- 6.711.2.2 `SendExecutor& activemq::cmsutil::CmsTemplate::SendExecutor::operator= (const SendExecutor &)` [inline, protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.712 decaf::net::ServerSocket Class Reference

This class implements server sockets.

`#include <src/main/decaf/net/ServerSocket.h>`

Inheritance diagram for `decaf::net::ServerSocket`:

Public Member Functions

- **ServerSocket** ()
Creates a non-bound server socket.
- **ServerSocket** (int port) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException)
*Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **ServerSocket** (int port, int backlog) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException)
*Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **ServerSocket** (int port, int backlog, const **InetAddress** *address) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException)
*Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- virtual ~**ServerSocket** ()
*Releases socket handle if **close()** (p. 3142) hasn't been called.*
- virtual void **bind** (const std::string &host, int port) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException)
Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.
- virtual void **bind** (const std::string &host, int port, int backlog) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException)
Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.
- virtual **Socket** * **accept** () throw (decaf::io::IOException)
*Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3136), the caller blocks until a connection is made.*
- virtual void **close** () throw (decaf::io::IOException)
Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.
- virtual bool **isClosed** () const
- virtual bool **isBound** () const
- virtual int **getReceiveBufferSize** () const throw (SocketException)
*Gets the receive buffer size for this socket, **SO_RCVBUF**.*
- virtual void **setReceiveBufferSize** (int size) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException)
*Sets the receive buffer size for this socket, **SO_RCVBUF**.*
- virtual bool **getReuseAddress** () const throw (SocketException)
*Gets the reuse address flag, **SO_REUSEADDR**.*

- virtual void **setReuseAddress** (bool reuse) throw (SocketException)
Sets the reuse address flag, SO_REUSEADDR.
- virtual int **getSoTimeout** () const throw (SocketException)
Gets the timeout for socket operations, SO_TIMEOUT.
- virtual void **setSoTimeout** (int timeout) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException)
Sets the timeout for socket operations, SO_TIMEOUT.
- virtual int **getLocalPort** () const
*Gets the port number on the Local machine that this **ServerSocket** (p. 3136) is bound to.*
- virtual std::string **toString** () const

Static Public Member Functions

- static void **setSocketImplFactory** (SocketImplFactory *factory) throw (decaf::io::IOException, decaf::net::SocketException)
*Sets the instance of a **SocketImplFactory** (p. 3314) that the **ServerSocket** (p. 3136) class should use when new instances of this class are created.*

Protected Member Functions

- **ServerSocket** (SocketImpl *impl)
*Creates a **ServerSocket** (p. 3136) wrapping the provided **SocketImpl** (p. 3306) instance, this **Socket** (p. 3281) is considered unconnected.*
- virtual void **implAccept** (Socket *socket) throw (decaf::io::IOException)
*Virtual method that allows a **ServerSocket** (p. 3136) subclass to override the accept call and provide its own **SocketImpl** (p. 3306) for the socket.*
- virtual int **getDefaultBacklog** ()
Allows a subclass to override what is considered the default backlog.
- void **checkClosed** () const throw (decaf::io::IOException)
- void **ensureCreated** () const throw (decaf::io::IOException)
- void **setupSocketImpl** (int port, int backlog, const InetAddress *ifAddress)

6.712.1 Detailed Description

This class implements server sockets. A server socket waits for requests to come in over the network.

The actual work of the server socket is performed by an instance of the **SocketImpl** (p. 3306) class. An application can change the socket factory that creates the socket implementation to configure itself to create sockets of a particular type.

Since

1.0

6.712.2 Constructor & Destructor Documentation

6.712.2.1 `decaf::net::ServerSocket::ServerSocket ()`

Creates a non-bound server socket.

6.712.2.2 `decaf::net::ServerSocket::ServerSocket (int port) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException)`

Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 3314) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3306) is created.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

Exceptions

IOException if there is an I/O error while performing this operation.

IllegalArgumentException if the port value is negative or greater than 65535.

6.712.2.3 `decaf::net::ServerSocket::ServerSocket (int port, int backlog) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException)`

Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at *backlog*, connections that arrive after the backlog has been reached are refused. If *backlog* is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3314) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3306) is created.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The the number of incoming connection attempts to queue before connections are refused.

Exceptions

IOException if there is an I/O error while performing this operation.

IllegalArgumentException if the port value is negative or greater than 65535.

6.712.2.4 decaf::net::ServerSocket::ServerSocket (int *port*, int *backlog*, const InetAddress * *address*) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException)

Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.

If the value of the *ifAddress* is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at *backlog*, connections that arrive after the backlog has been reached are refused. If *backlog* is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3314) is registered then the *createSocketImpl* method on the factory will be called otherwise a default **SocketImpl** (p. 3306) is created.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The the number of incoming connection attempts to queue before connections are refused.

ifAddress The IP Address to bind to on the local machine.

Exceptions

IOException if there is an I/O error while performing this operation.

IllegalArgumentException if the port value is negative or greater than 65535.

6.712.2.5 virtual decaf::net::ServerSocket::~~ServerSocket () [virtual]

Releases socket handle if *close()* (p. 3142) hasn't been called.

6.712.2.6 decaf::net::ServerSocket::ServerSocket (SocketImpl * *impl*) [protected]

Creates a **ServerSocket** (p. 3136) wrapping the provided **SocketImpl** (p. 3306) instance, this **Socket** (p. 3281) is considered unconnected.

The **ServerSocket** (p. 3136) class takes ownership of this **SocketImpl** (p. 3306) pointer and will delete it when the **Socket** (p. 3281) class is destroyed.

Parameters

impl The **SocketImpl** (p. 3306) instance to wrap.

Exceptions

NullPointerException if the passed **SocketImpl** (p. 3306) is Null.

6.712.3 Member Function Documentation

6.712.3.1 virtual Socket* decaf::net::ServerSocket::accept () throw (decaf::io::IOException) [virtual]

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3136), the caller blocks until a connection is made.

If the SO_TIMEOUT option is set this method could throw a **SocketTimeoutException** (p. 3319) if the operation times out.

Returns

a new **Socket** (p. 3281) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

Exceptions

IOException if an I/O error occurs while binding the socket.

SocketException (p. 3298) if an error occurs while blocking on the accept call.

SocketTimeoutException (p. 3319) if the SO_TIMEOUT option was used and the accept timed out.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2666).

6.712.3.2 virtual void decaf::net::ServerSocket::bind (const std::string & host, int port, int backlog) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException) [virtual]

Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.

If the backlog is greater than zero it will be used instead of the default value, otherwise the default value is used and no error is generated.

Parameters

host The IP address or host name.

port The TCP port between 1..65535.

backlog The size of listen backlog.

Exceptions

IOException if an I/O error occurs while binding the socket.

IllegalArgumentException if the parameters are not valid.

6.712.3.3 virtual void decaf::net::ServerSocket::bind (const std::string & host, int port) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException) [virtual]

Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.

Parameters

host The IP address or host name.

port The TCP port between 1..65535.

Exceptions

IOException if an I/O error occurs while binding the socket.

IllegalArgumentException if the parameters are not valid.

6.712.3.4 void decaf::net::ServerSocket::checkClosed () const throw (decaf::io::IOException) [protected]

6.712.3.5 virtual void decaf::net::ServerSocket::close () throw (decaf::io::IOException) [virtual]

Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.

Exceptions

IOException if an I/O error occurs while performing this operation.

6.712.3.6 void decaf::net::ServerSocket::ensureCreated () const throw (decaf::io::IOException) [protected]

6.712.3.7 virtual int decaf::net::ServerSocket::getDefaultBacklog () [protected, virtual]

Allows a subclass to override what is considered the default backlog.

Returns

the default backlog for connections.

6.712.3.8 virtual int decaf::net::ServerSocket::getLocalPort () const [virtual]

Gets the port number on the Local machine that this **ServerSocket** (p. 3136) is bound to.

Returns

the port number of this machine that is bound, if not bound returns -1.

6.712.3.9 virtual int decaf::net::ServerSocket::getReceiveBufferSize () const throw (SocketException) [virtual]

Gets the receive buffer size for this socket, SO_RCVBUF.

This is the buffer used by the underlying platform socket to buffer received data.

Returns

the receive buffer size in bytes.

Exceptions

SocketException (p. 3298) if the operation fails.

6.712.3.10 `virtual bool decaf::net::ServerSocket::getReuseAddress () const
 throw (SocketException) [virtual]`

Gets the reuse address flag, SO_REUSEADDR.

Returns

True if the address can be reused.

Exceptions

SocketException (p. 3298) if the operation fails.

6.712.3.11 `virtual int decaf::net::ServerSocket::getSoTimeout () const throw (
 SocketException) [virtual]`

Gets the timeout for socket operations, SO_TIMEOUT.

Returns

The timeout in milliseconds for socket operations.

Exceptions

SocketException (p. 3298) Thrown if unable to retrieve the information.

6.712.3.12 `virtual void decaf::net::ServerSocket::implAccept (Socket * socket)
 throw (decaf::io::IOException) [protected, virtual]`

Virtual method that allows a **ServerSocket** (p. 3136) subclass to override the accept call and provide its own **SocketImpl** (p. 3306) for the socket.

Parameters

socket The socket object whose **SocketImpl** (p. 3306) should be used for the accept call.

Exceptions

IOException if an I/O error occurs while performing this operation.

6.712.3.13 `virtual bool decaf::net::ServerSocket::isBound () const [virtual]`

Returns

true of the server socket is bound.

6.712.3.14 `virtual bool decaf::net::ServerSocket::isClosed () const [virtual]`

Returns

true if the close method has been called on the **ServerSocket** (p. 3136).

6.712.3.15 `virtual void decaf::net::ServerSocket::setReceiveBufferSize (int size) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException) [virtual]`

Sets the receive buffer size for this socket, SO_RCVBUF.

Parameters

size Number of bytes to set the receive buffer to.

Exceptions

SocketException (p. 3298) if the operation fails.

IllegalArgumentException if the value is zero or negative.

6.712.3.16 `virtual void decaf::net::ServerSocket::setReuseAddress (bool reuse) throw (SocketException) [virtual]`

Sets the reuse address flag, SO_REUSEADDR.

Parameters

reuse If true, sets the flag.

Exceptions

SocketException (p. 3298) if the operation fails.

6.712.3.17 `static void decaf::net::ServerSocket::setSocketImplFactory (SocketImplFactory * factory) throw (decaf::io::IOException, decaf::net::SocketException) [static]`

Sets the instance of a **SocketImplFactory** (p. 3314) that the **ServerSocket** (p. 3136) class should use when new instances of this class are created.

This method is only allowed to be used once during the lifetime of the application.

Parameters

factory The instance of a **SocketImplFactory** (p. 3314) to use when new **SocketImpl** (p. 3306) objects are created.

Exceptions

IOException if an I/O error occurs while performing this operation.

SocketException (p. 3298) if this method has already been called with a valid factory.

6.712.3.18 `virtual void decaf::net::ServerSocket::setSoTimeout (int timeout) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException) [virtual]`

Sets the timeout for socket operations, `SO_TIMEOUT`.

A value of zero indicates that timeout is infinite for operations on this socket.

Parameters

timeout The timeout in milliseconds for socket operations.

Exceptions

SocketException (p. 3298) Thrown if unable to set the information.

IllegalArgumentException if the timeout value is negative.

6.712.3.19 `void decaf::net::ServerSocket::setupSocketImpl (int port, int backlog, const InetAddress * ifAddress) [protected]`

6.712.3.20 `virtual std::string decaf::net::ServerSocket::toString () const [virtual]`

Returns

a string representing this `ServerSocket` (p. 3136).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ServerSocket.h`

6.713 decaf::net::ServerSocketFactory Class Reference

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

```
#include <src/main/decaf/net/ServerSocketFactory.h>
```

Inheritance diagram for `decaf::net::ServerSocketFactory`:

Public Member Functions

- `virtual ~ServerSocketFactory ()`
- `virtual ServerSocket * createServerSocket ()`
Create a new `ServerSocket` (p. 3136) that is unbound.
- `virtual ServerSocket * createServerSocket (int port)=0`
Create a new `ServerSocket` (p. 3136) that is bound to the given port.
- `virtual ServerSocket * createServerSocket (int port, int backlog)=0`

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

- virtual **ServerSocket** * **createServerSocket** (int port, int backlog, const **InetAddress** *address)=0

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

Static Public Member Functions

- static **ServerSocketFactory** * **getDefault** ()

Returns the Default **ServerSocket** (p. 3136) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller.

Protected Member Functions

- **ServerSocketFactory** ()

6.713.1 Detailed Description

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

Since

1.0

6.713.2 Constructor & Destructor Documentation

6.713.2.1 decaf::net::ServerSocketFactory::ServerSocketFactory () [protected]

6.713.2.2 virtual decaf::net::ServerSocketFactory::~~ServerSocketFactory () [virtual]

6.713.3 Member Function Documentation

6.713.3.1 virtual **ServerSocket*** decaf::net::ServerSocketFactory::createServerSocket () [virtual]

Create a new **ServerSocket** (p. 3136) that is unbound.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Reimplemented in decaf::internal::net::DefaultServerSocketFactory (p. 1575), decaf::internal::net::ssl::DefaultSSLServerSocketFactory (p. 1584), and decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory (p. 2671).

6.713.3.2 virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket (int *port*) [pure virtual]

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1576), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1585), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2671).

6.713.3.3 virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket (int *port*, int *backlog*, const InetAddress * *address*) [pure virtual]

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3136) will listen on all interfaces.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

address The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1576), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1585), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2672).

6.713.3.4 virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket (int *port*, int *backlog*) [pure virtual]

Create a new **ServerSocket** (p. 3136) that is bound to the given port.

The **ServerSocket** (p. 3136) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3136) will use the specified connection backlog setting.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The number of pending connect request the **ServerSocket** (p. 3136) can queue.

Returns

new **ServerSocket** (p. 3136) pointer that is owned by the caller.

Exceptions

IOException if the **ServerSocket** (p. 3136) cannot be created for some reason.

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1576), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1585), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2672).

6.713.3.5 static ServerSocketFactory* decaf::net::ServerSocketFactory::getDefault () [static]

Returns the Default **ServerSocket** (p. 3136) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller.

Only one default **ServerSocketFactory** (p. 3145) exists for the lifetime of the Application.

Returns

the default **ServerSocketFactory** (p. 3145) for this application.

Reimplemented in **decaf::net::ssl::SSLServerSocketFactory** (p. 3336).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ServerSocketFactory.h**

6.714 cms::Session Class Reference

A **Session** (p. 3148) object is a single-threaded context for producing and consuming messages.

```
#include <src/main/cms/Session.h>
```

Inheritance diagram for cms::Session:

Public Types

- enum **AcknowledgeMode** {
AUTO_ACKNOWLEDGE, **DUPS_OK_ACKNOWLEDGE**, **CLIENT_**
ACKNOWLEDGE, **SESSION_TRANSACTIONED**,
INDIVIDUAL_ACKNOWLEDGE }

Public Member Functions

- virtual `~Session ()`
- virtual void `close ()=0 throw (CMSEException)`
Closes this session as well as any active child consumers or producers.
- virtual void `commit ()=0 throw (CMSEException)`
Commits all messages done in this transaction and releases any locks currently held.
- virtual void `rollback ()=0 throw (CMSEException)`
Rolls back all messages done in this transaction and releases any locks currently held.
- virtual void `recover ()=0 throw (CMSEException)`
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual `MessageConsumer * createConsumer (const Destination *destination)=0 throw (CMSEException)`
*Creates a **MessageConsumer** (p. 2423) for the specified destination.*
- virtual `MessageConsumer * createConsumer (const Destination *destination, const std::string &selector)=0 throw (CMSEException)`
*Creates a **MessageConsumer** (p. 2423) for the specified destination, using a message selector.*
- virtual `MessageConsumer * createConsumer (const Destination *destination, const std::string &selector, bool noLocal)=0 throw (CMSEException)`
*Creates a **MessageConsumer** (p. 2423) for the specified destination, using a message selector.*
- virtual `MessageConsumer * createDurableConsumer (const Topic *destination, const std::string &name, const std::string &selector, bool noLocal=false)=0 throw (CMSEException)`
*Creates a durable subscriber to the specified topic, using a **Message** (p. 2375) selector.*
- virtual `MessageProducer * createProducer (const Destination *destination)=0 throw (CMSEException)`
*Creates a **MessageProducer** (p. 2550) to send messages to the specified destination.*
- virtual `QueueBrowser * createBrowser (const cms::Queue *queue)=0 throw (CMSEException)`
*Creates a new **QueueBrowser** (p. 2951) to peek at Messages on the given **Queue** (p. 2947).*
- virtual `QueueBrowser * createBrowser (const cms::Queue *queue, const std::string &selector)=0 throw (CMSEException)`
*Creates a new **QueueBrowser** (p. 2951) to peek at Messages on the given **Queue** (p. 2947).*
- virtual `Queue * createQueue (const std::string &queueName)=0 throw (CMSEException)`
*Creates a queue identity given a **Queue** (p. 2947) name.*
- virtual `Topic * createTopic (const std::string &topicName)=0 throw (CMSEException)`
*Creates a topic identity given a **Queue** (p. 2947) name.*

- virtual **TemporaryQueue** * **createTemporaryQueue** ()=0 throw (CMSEException)
*Creates a **TemporaryQueue** (p. 3516) object.*
- virtual **TemporaryTopic** * **createTemporaryTopic** ()=0 throw (CMSEException)
*Creates a **TemporaryTopic** (p. 3517) object.*
- virtual **Message** * **createMessage** ()=0 throw (CMSEException)
*Creates a new **Message** (p. 2375).*
- virtual **BytesMessage** * **createBytesMessage** ()=0 throw (CMSEException)
*Creates a **BytesMessage** (p. 979).*
- virtual **BytesMessage** * **createBytesMessage** (const unsigned char *bytes, int bytes-Size)=0 throw (CMSEException)
*Creates a **BytesMessage** (p. 979) and sets the payload to the passed value.*
- virtual **StreamMessage** * **createStreamMessage** ()=0 throw (CMSEException)
*Creates a new **StreamMessage** (p. 3415).*
- virtual **TextMessage** * **createTextMessage** ()=0 throw (CMSEException)
*Creates a new **TextMessage** (p. 3519).*
- virtual **TextMessage** * **createTextMessage** (const std::string &text)=0 throw (CMSEException)
*Creates a new **TextMessage** (p. 3519) and set the text to the value given.*
- virtual **MapMessage** * **createMapMessage** ()=0 throw (CMSEException)
*Creates a new **MapMessage** (p. 2318).*
- virtual **AcknowledgeMode** **getAcknowledgeMode** () const =0 throw (CMSEException)
Returns the acknowledgment mode of the session.
- virtual bool **isTransacted** () const =0 throw (CMSEException)
*Gets if the Sessions is a Transacted **Session** (p. 3148).*
- virtual void **unsubscribe** (const std::string &name)=0 throw (CMSEException)
Unsubscribes a durable subscription that has been created by a client.

6.714.1 Detailed Description

A **Session** (p.3148) object is a single-threaded context for producing and consuming messages. A session serves several purposes:

- It is a factory for its message producers and consumers.
- It supplies provider-optimized message factories.
- It is a factory for **TemporaryTopics** and **TemporaryQueues**.

- It provides a way to create **Queue** (p. 2947) or **Topic** (p. 3568) objects for those clients that need to dynamically manipulate provider-specific destination names.
- It supports a single series of transactions that combine work spanning its producers and consumers into atomic units.
- It defines a serial order for the messages it consumes and the messages it produces.
- It retains messages it consumes until they have been acknowledged.
- It serializes execution of message listeners registered with its message consumers.

A session can create and service multiple message producers and consumers.

One typical use is to have a thread block on a synchronous **MessageConsumer** (p. 2423) until a message arrives. The thread may then use one or more of the Session's MessageProducers.

If a client desires to have one thread produce messages while others consume them, the client should use a separate session for its producing thread.

Certain rules apply to a session's `close` method and are detailed below.

- There is no need to close the producers and consumers of a closed session.
- The close call will block until a receive call or message listener in progress has completed. A blocked message consumer receive call returns null when this session is closed.
- Closing a transacted session must roll back the transaction in progress.
- The close method is the only **Session** (p. 3148) method that can be called concurrently.
- Invoking any other **Session** (p. 3148) method on a closed session must throw an **IllegalStateException** (p. 1868). Closing a closed session must not throw any exceptions.

Transacted Sessions

When a **Session** (p. 3148) is created it can be set to operate in a Transaction based mode. Each **Session** (p. 3148) then operates in a single transaction for all Producers and Consumers of that **Session** (p. 3148). Messages sent and received within a Transaction are grouped into an atomic unit that is committed or rolled back together.

For a **MessageProducer** (p. 2550) this implies that all messages sent by the producer are not sent to the Provider unit the commit call is made. Rolling back the Transaction results in all produced Messages being dropped.

For a **MessageConsumer** (p. 2423) this implies that all received messages are not Acknowledged until the Commit call is made. Rolling back the Transaction results in all Consumed **Message** (p. 2375) being redelivered to the client, the Provider may allow configuration that limits the Maximum number of redeliveries for a **Message** (p. 2375).

Since

1.0

6.714.2 Member Enumeration Documentation

6.714.2.1 enum cms::Session::AcknowledgeMode

Enumerator:

AUTO_ACKNOWLEDGE With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully

returned from a call to receive or when the message listener the session has called to process the message successfully returns.

DUPS_OK_ACKNOWLEDGE With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns. Acknowledgments may be delayed in this mode to increase performance at the cost of the message being redelivered this client fails.

CLIENT_ACKNOWLEDGE With this acknowledgment mode, the client acknowledges a consumed message by calling the message's acknowledge method.

SESSION_TRANSACTED Messages will be consumed when the transaction commits.

INDIVIDUAL_ACKNOWLEDGE **Message** (p. 2375) will be acknowledged individually. Normally the acks sent acknowledge the given message and all messages received before it, this mode only acknowledges one message.

6.714.3 Constructor & Destructor Documentation

6.714.3.1 `virtual cms::Session::~~Session () [inline, virtual]`

6.714.4 Member Function Documentation

6.714.4.1 `virtual void cms::Session::close () throw (CMSEException) [pure virtual]`

Closes this session as well as any active child consumers or producers.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implements **cms::Closeable** (p. 1065).

Implemented in **activemq::cmsutil::PooledSession** (p. 2768).

6.714.4.2 `virtual void cms::Session::commit () throw (CMSEException) [pure virtual]`

Commits all messages done in this transaction and releases any locks currently held.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

IllegalStateException (p. 1868) - if the method is not called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2768).

Referenced by **activemq::cmsutil::PooledSession::commit()**.

6.714.4.3 `virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue * queue) throw (CMSEException) [pure virtual]`

Creates a new **QueueBrowser** (p. 2951) to peek at Messages on the given **Queue** (p. 2947).

Parameters

queue the **Queue** (p. 2947) to browse

Returns

New **QueueBrowser** (p. 2951) that is owned by the caller.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

InvalidDestinationException (p. 1993) - if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2768).

6.714.4.4 `virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue
* queue, const std::string & selector) throw (CMSEException) [pure
virtual]`

Creates a new **QueueBrowser** (p. 2951) to peek at Messages on the given **Queue** (p. 2947).

Parameters

queue the **Queue** (p. 2947) to browse

selector the **Message** (p. 2375) selector to filter which messages are browsed.

Returns

New **QueueBrowser** (p. 2951) that is owned by the caller.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

InvalidDestinationException (p. 1993) - if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2768).

6.714.4.5 `virtual BytesMessage* cms::Session::createBytesMessage () throw (CMSEException) [pure virtual]`

Creates a **BytesMessage** (p. 979).

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2769).

6.714.4.6 `virtual BytesMessage* cms::Session::createBytesMessage (const
unsigned char * bytes, int bytesSize) throw (CMSEException) [pure
virtual]`

Creates a **BytesMessage** (p. 979) and sets the payload to the passed value.

Parameters

bytes an array of bytes to set in the message

bytesSize the size of the bytes array, or number of bytes to use

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2769).

6.714.4.7 virtual `MessageConsumer* cms::Session::createConsumer (const Destination * destination) throw (CMSEException)` [pure virtual]

Creates a **MessageConsumer** (p. 2423) for the specified destination.

Parameters

destination the **Destination** (p. 1610) that this consumer receiving messages for.

Returns

pointer to a new **MessageConsumer** (p. 2423) that is owned by the caller (caller deletes)

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

InvalidDestinationException (p. 1993) - if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 2770).

6.714.4.8 virtual `MessageConsumer* cms::Session::createConsumer (const Destination * destination, const std::string & selector) throw (CMSEException)` [pure virtual]

Creates a **MessageConsumer** (p. 2423) for the specified destination, using a message selector.

Parameters

destination the **Destination** (p. 1610) that this consumer receiving messages for.

selector the **Message** (p. 2375) Selector to use

Returns

pointer to a new **MessageConsumer** (p. 2423) that is owned by the caller (caller deletes)

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

InvalidDestinationException (p. 1993) - if an invalid destination is specified.

InvalidSelectorException (p. 1999) - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2771).

6.714.4.9 `virtual MessageConsumer* cms::Session::createConsumer (const Destination * destination, const std::string & selector, bool noLocal) throw (CMSException) [pure virtual]`

Creates a **MessageConsumer** (p. 2423) for the specified destination, using a message selector.

Parameters

destination the **Destination** (p. 1610) that this consumer receiving messages for.

selector the **Message** (p. 2375) Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new **MessageConsumer** (p. 2423) that is owned by the caller (caller deletes)

Exceptions

CMSException (p. 1074) - If an internal error occurs.

InvalidDestinationException (p. 1993) - if an invalid destination is specified.

InvalidSelectorException (p. 1999) - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2771).

6.714.4.10 `virtual MessageConsumer* cms::Session::createDurableConsumer (const Topic * destination, const std::string & name, const std::string & selector, bool noLocal = false) throw (CMSException) [pure virtual]`

Creates a durable subscriber to the specified topic, using a **Message** (p. 2375) selector.

Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

Parameters

destination the topic to subscribe to

name The name used to identify the subscription

selector the **Message** (p. 2375) Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new durable **MessageConsumer** (p. 2423) that is owned by the caller (caller deletes)

Exceptions

CMSException (p. 1074) - If an internal error occurs.

InvalidDestinationException (p. 1993) - if an invalid destination is specified.

InvalidSelectorException (p. 1999) - if the message selector is invalid.

Implemented in `activemq::cmsutil::PooledSession` (p. 2772).

6.714.4.11 `virtual MapMessage* cms::Session::createMapMessage () throw (CMSEException) [pure virtual]`

Creates a new `MapMessage` (p. 2318).

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in `activemq::cmsutil::PooledSession` (p. 2772).

6.714.4.12 `virtual Message* cms::Session::createMessage () throw (CMSEException) [pure virtual]`

Creates a new `Message` (p. 2375).

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in `activemq::cmsutil::PooledSession` (p. 2772).

6.714.4.13 `virtual MessageProducer* cms::Session::createProducer (const Destination * destination) throw (CMSEException) [pure virtual]`

Creates a `MessageProducer` (p. 2550) to send messages to the specified destination.

Parameters

destination the `Destination` (p. 1610) to send on

Returns

New `MessageProducer` (p. 2550) that is owned by the caller.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

InvalidDestinationException (p. 1993) - if an invalid destination is specified.

Implemented in `activemq::cmsutil::PooledSession` (p. 2773).

6.714.4.14 `virtual Queue* cms::Session::createQueue (const std::string & queueName) throw (CMSEException) [pure virtual]`

Creates a queue identity given a `Queue` (p. 2947) name.

Parameters

queueName the name of the new **Queue** (p. 2947)

Returns

new **Queue** (p. 2947) pointer that is owned by the caller.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2773).

6.714.4.15 `virtual StreamMessage* cms::Session::createStreamMessage () throw (CMSEException) [pure virtual]`

Creates a new **StreamMessage** (p. 3415).

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2773).

6.714.4.16 `virtual TemporaryQueue* cms::Session::createTemporaryQueue () throw (CMSEException) [pure virtual]`

Creates a **TemporaryQueue** (p. 3516) object.

Returns

new **TemporaryQueue** (p. 3516) pointer that is owned by the caller.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2774).

6.714.4.17 `virtual TemporaryTopic* cms::Session::createTemporaryTopic () throw (CMSEException) [pure virtual]`

Creates a **TemporaryTopic** (p. 3517) object.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2774).

6.714.4.18 `virtual TextMessage* cms::Session::createTextMessage (const std::string & text) throw (CMSEException) [pure virtual]`

Creates a new **TextMessage** (p. 3519) and set the text to the value given.

Parameters

text the initial text for the message

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2774).

6.714.4.19 `virtual TextMessage* cms::Session::createTextMessage () throw (CMSEException) [pure virtual]`

Creates a new **TextMessage** (p. 3519).

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2774).

6.714.4.20 `virtual Topic* cms::Session::createTopic (const std::string & topicName) throw (CMSEException) [pure virtual]`

Creates a topic identity given a **Queue** (p. 2947) name.

Parameters

topicName the name of the new **Topic** (p. 3568)

Returns

new **Topic** (p. 3568) pointer that is owned by the caller.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2775).

6.714.4.21 `virtual AcknowledgeMode cms::Session::getAcknowledgeMode () const throw (CMSEException) [pure virtual]`

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2775).

6.714.4.22 `virtual bool cms::Session::isTransacted () const throw (CMSEException) [pure virtual]`

Gets if the Sessions is a Transacted **Session** (p. 3148).

Returns

transacted true - false.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2776).

6.714.4.23 `virtual void cms::Session::recover () throw (CMSEException) [pure virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions

CMSEException (p. 1074) - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException (p. 1868) - if the method is called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2776).

6.714.4.24 `virtual void cms::Session::rollback () throw (CMSEException) [pure virtual]`

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

IllegalStateException (p. 1868) - if the method is not called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2776).

6.714.4.25 `virtual void cms::Session::unsubscribe (const std::string & name)
throw (CMSEException) [pure virtual]`

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active **MessageConsumer** (p. 2423) or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters

name The name used to identify this subscription

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2777).

The documentation for this class was generated from the following file:

- `src/main/cms/Session.h`

6.715 activemq::cmsutil::SessionCallback Class Reference

Callback for executing any number of operations on a provided CMS Session.

```
#include <src/main/activemq/cmsutil/SessionCallback.h>
```

Inheritance diagram for `activemq::cmsutil::SessionCallback`:

Public Member Functions

- `virtual ~SessionCallback ()`
- `virtual void doInCms (cms::Session *session)=0 throw (cms::CMSEException)`

Execute any number of operations against the supplied CMS session.

6.715.1 Detailed Description

Callback for executing any number of operations on a provided CMS Session.

6.715.2 Constructor & Destructor Documentation

6.715.2.1 `virtual activemq::cmsutil::SessionCallback::~SessionCallback ()`
`[inline, virtual]`

6.715.3 Member Function Documentation

6.715.3.1 `virtual void activemq::cmsutil::SessionCallback::doInCms (cms::Session * session) throw (cms::CMSException)` `[pure virtual]`

Execute any number of operations against the supplied CMS session.

Parameters

session the CMS Session

Exceptions

cms::CMSException (p. 1074) if thrown by CMS API methods

Implemented in `activemq::cmsutil::CmsTemplate::ProducerExecutor` (p. 2868), and `activemq::cmsutil::CmsTemplate::ReceiveExecutor` (p. 2971).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/SessionCallback.h`

6.716 activemq::commands::SessionId Class Reference

```
#include <src/main/activemq/commands/SessionId.h>
```

Inheritance diagram for `activemq::commands::SessionId`:

Public Types

- `typedef decaf::lang::PointerComparator< SessionId > COMPARATOR`

Public Member Functions

- `SessionId ()`
- `SessionId (const SessionId &other)`
- `SessionId (const ConnectionId *connectionId, long long sessionId)`
- `SessionId (const ProducerId *producerId)`
- `SessionId (const ConsumerId *consumerId)`
- `virtual ~SessionId ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual SessionId * cloneDataStructure () const`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*

- const **Pointer**< **ConnectionId** > & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual int **compareTo** (const **SessionId** &value) const
- virtual bool **equals** (const **SessionId** &value) const
- virtual bool **operator==** (const **SessionId** &value) const
- virtual bool **operator<** (const **SessionId** &value) const
- **SessionId** & **operator=** (const **SessionId** &other)

Static Public Attributes

- static const unsigned char **ID_SESSIONID** = 121

Protected Attributes

- std::string **connectionId**
- long long **value**

6.716.1 Member Typedef Documentation

6.716.1.1 `typedef decaf::lang::PointerComparator<SessionId>
activemq::commands::SessionId::COMPARATOR`

6.716.2 Constructor & Destructor Documentation

6.716.2.1 `activemq::commands::SessionId::SessionId ()`

6.716.2.2 `activemq::commands::SessionId::SessionId (const SessionId & other)`

6.716.2.3 `activemq::commands::SessionId::SessionId (const ConnectionId *
connectionId, long long sessionId)`

6.716.2.4 `activemq::commands::SessionId::SessionId (const ProducerId *
producerId)`

6.716.2.5 `activemq::commands::SessionId::SessionId (const ConsumerId *
consumerId)`

6.716.2.6 `virtual activemq::commands::SessionId::~~SessionId () [virtual]`

6.716.3 Member Function Documentation

6.716.3.1 `virtual SessionId* activemq::commands::SessionId::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.716.3.2 `virtual int activemq::commands::SessionId::compareTo (const SessionId
& value) const [virtual]`

6.716.3.3 `virtual void activemq::commands::SessionId::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.716.3.4 `virtual bool activemq::commands::SessionId::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.716.3.5 virtual bool activemq::commands::SessionId::equals (const SessionId & *value*) const [virtual]

6.716.3.6 virtual std::string& activemq::commands::SessionId::getConnectionId () [virtual]

6.716.3.7 virtual const std::string& activemq::commands::SessionId::getConnectionId () const [virtual]

6.716.3.8 virtual unsigned char activemq::commands::SessionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

6.716.3.9 const Pointer<ConnectionId>& activemq::commands::SessionId::getParentId () const

6.716.3.10 virtual long long activemq::commands::SessionId::getValue () const [virtual]

6.716.3.11 virtual bool activemq::commands::SessionId::operator< (const SessionId & *value*) const [virtual]

6.716.3.12 SessionId& activemq::commands::SessionId::operator= (const SessionId & *other*)

6.716.3.13 virtual bool activemq::commands::SessionId::operator== (const SessionId & *value*) const [virtual]

6.716.3.14 virtual void activemq::commands::SessionId::setConnectionId (const std::string & *connectionId*) [virtual]

6.716.3.15 virtual void activemq::commands::SessionId::setValue (long long *value*) [virtual]

6.716.3.16 virtual std::string activemq::commands::SessionId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.716.4 Field Documentation

6.716.4.1 `std::string activemq::commands::SessionId::connectionId` [protected]

6.716.4.2 `const unsigned char activemq::commands::SessionId::ID_SESSIONID = 121` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.716.4.3 `long long activemq::commands::SessionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionId.h`

6.717 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3165).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller`:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.717.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3165). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.717.2 Constructor & Destructor Documentation

6.717.2.1 `activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::SessionIdMarshaller ()` [inline]

6.717.2.2 `virtual activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::~~SessionIdMarshaller ()` [inline, virtual]

6.717.3 Member Function Documentation

6.717.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::createObject () const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.717.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::getDataStructureType () const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.717.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.717.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.717.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.717.3.6 virtual void `activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal2`
(`OpenWireFormat * wireFormat`, `commands::DataStructure`
* `dataStructure`, `decaf::io::DataOutputStream * dataOut`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.717.3.7 virtual void `activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightUnmarshal`
(`OpenWireFormat * wireFormat`, `commands::DataStructure`
* `dataStructure`, `decaf::io::DataInputStream * dataIn`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h`

6.718 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3169).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller**:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.718.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3169). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.718.2 Constructor & Destructor Documentation

6.718.2.1 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::SessionIdMarshaller () [inline]

6.718.2.2 virtual
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::~~SessionIdMarshaller () [inline, virtual]

6.718.3 Member Function Documentation

6.718.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.718.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.718.3.3 virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.718.3.4 virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.718.3.5 virtual int activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.718.3.6 virtual void **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal2**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.718.3.7 virtual void **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h`

6.719 **activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3173).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller**:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.719.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3173). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.719.2 Constructor & Destructor Documentation

6.719.2.1 `activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::SessionIdMarshaller () [inline]`

6.719.2.2 `virtual
activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::~~SessionIdMarshaller () [inline, virtual]`

6.719.3 Member Function Documentation

6.719.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.719.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::getDataStructureType
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.719.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

```
6.719.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

```
6.719.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.719.3.6 virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.719.3.7 virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h

6.720 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3176).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller`:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.720.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3176). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.720.2 Constructor & Destructor Documentation

6.720.2.1 `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::SessionIdMarshaller () [inline]`

6.720.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::~~SessionIdMarshaller
() [inline, virtual]`

6.720.3 Member Function Documentation

6.720.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.720.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::getDataStructureType
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.720.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.720.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.720.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.720.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.720.3.7 virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightUnmarshal
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h`

6.721 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3180).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller**:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.721.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3180). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.721.2 Constructor & Destructor Documentation

6.721.2.1 `activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::SessionIdMarshaller () [inline]`

6.721.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::~~SessionIdMarshaller
() [inline, virtual]`

6.721.3 Member Function Documentation

6.721.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.721.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::getDataStructureType
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.721.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.721.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.721.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.721.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.721.3.7 virtual void `activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightUnmarshal`
(`OpenWireFormat` * *wireFormat*, `commands::DataStructure`
* *dataStructure*, `decaf::io::DataInputStream` * *dataIn*,
`utils::BooleanStream` * *bs*) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h`

6.722 `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `SessionIdMarshaller` (p. 3184).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller`:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.722.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3184). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.722.2 Constructor & Destructor Documentation

6.722.2.1 `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::SessionIdMarshaller () [inline]`

6.722.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::~~SessionIdMarshaller () [inline, virtual]`

6.722.3 Member Function Documentation

6.722.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.722.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.722.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.722.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.722.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.722.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.722.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h

6.723 activemq::commands::SessionInfo Class Reference

```
#include <src/main/activemq/commands/SessionInfo.h>
```

Inheritance diagram for **activemq::commands::SessionInfo**:

Public Member Functions

- **SessionInfo** ()
- virtual **~SessionInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **SessionInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- unsigned int **getAckMode** () const
- void **setAckMode** (unsigned int mode)
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer< SessionId > & getSessionId** () const
- virtual **Pointer< SessionId > & getSessionId** ()
- virtual void **setSessionId** (const **Pointer< SessionId > &sessionId**)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_SESSIONINFO** = 4

Protected Attributes

- **Pointer< SessionId > sessionId**

6.723.1 Constructor & Destructor Documentation

6.723.1.1 activemq::commands::SessionInfo::SessionInfo ()

6.723.1.2 virtual activemq::commands::SessionInfo::~~SessionInfo () [virtual]

6.723.2 Member Function Documentation

6.723.2.1 virtual SessionInfo* activemq::commands::SessionInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.723.2.2 `virtual void activemq::commands::SessionInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.723.2.3 `Pointer<RemoveInfo> activemq::commands::SessionInfo::createRemoveCommand () const`

6.723.2.4 `virtual bool activemq::commands::SessionInfo::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 696).

6.723.2.5 `unsigned int activemq::commands::SessionInfo::getAckMode () const [inline]`

6.723.2.6 `virtual unsigned char activemq::commands::SessionInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.723.2.7** `virtual const Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId () const`
[virtual]
- 6.723.2.8** `virtual Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId ()`
[virtual]
- 6.723.2.9** `void activemq::commands::SessionInfo::setAckMode (unsigned int mode)` [inline]
- 6.723.2.10** `virtual void activemq::commands::SessionInfo::setSessionId (const Pointer< SessionId > & sessionId)` [virtual]
- 6.723.2.11** `virtual std::string activemq::commands::SessionInfo::toString () const`
[virtual]

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

- 6.723.2.12** `virtual Pointer<Command> activemq::commands::SessionInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1112).

6.723.3 Field Documentation

- 6.723.3.1** `const unsigned char activemq::commands::SessionInfo::ID_SESSIONINFO = 4` [static]
- 6.723.3.2** `Pointer<SessionId> activemq::commands::SessionInfo::sessionId`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionInfo.h`

6.724 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3192).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.724.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3192). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.724.2 Constructor & Destructor Documentation

6.724.2.1 `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::SessionInfoMarshaller () [inline]`

6.724.2.2 `virtual activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::~~SessionInfoMarshaller () [inline, virtual]`

6.724.3 Member Function Documentation

6.724.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.724.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.724.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

6.724.3.4 virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 730).

6.724.3.5 virtual int activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 731).

6.724.3.6 virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 732).

```
6.724.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h`

6.725 **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3195).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller**:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.725.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.3195). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.725.2 Constructor & Destructor Documentation

6.725.2.1 `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::SessionInfoMarshaller () [inline]`

6.725.2.2 `virtual activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::~~SessionInfoMarshaller () [inline, virtual]`

6.725.3 Member Function Documentation

6.725.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.725.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.725.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).


```

6.725.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

```

6.725.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```

6.725.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 726).

```
6.725.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h`

6.726 **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3199).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller**:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.726.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.3199). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.726.2 Constructor & Destructor Documentation

6.726.2.1 `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::SessionInfoMarshaller () [inline]`

6.726.2.2 `virtual activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::~~SessionInfoMarshaller () [inline, virtual]`

6.726.3 Member Function Documentation

6.726.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.726.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.726.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

```

6.726.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

```

6.726.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

```

6.726.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 719).

```
6.726.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h`

6.727 **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3203).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller**:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.727.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.3203). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.727.2 Constructor & Destructor Documentation

6.727.2.1 `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::SessionInfoMarshaller () [inline]`

6.727.2.2 `virtual activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::~~SessionInfoMarshaller () [inline, virtual]`

6.727.3 Member Function Documentation

6.727.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.727.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.727.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.727.3.4 virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 703).

6.727.3.5 virtual int activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 704).

6.727.3.6 virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 706).

```
6.727.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h`

6.728 **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3207).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller**:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.728.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3207). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.728.2 Constructor & Destructor Documentation

6.728.2.1 `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::SessionInfoMarshaller () [inline]`

6.728.2.2 `virtual activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::~~SessionInfoMarshaller () [inline, virtual]`

6.728.3 Member Function Documentation

6.728.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.728.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.728.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.728.3.4 virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 737).

6.728.3.5 virtual int activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 738).

6.728.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 739).

```
6.728.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h`

6.729 **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3211).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller**:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.729.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.3211). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.729.2 Constructor & Destructor Documentation

6.729.2.1 `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::SessionInfoMarshaller () [inline]`

6.729.2.2 `virtual activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::~~SessionInfoMarshaller () [inline, virtual]`

6.729.3 Member Function Documentation

6.729.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.729.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.729.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

6.729.3.4 virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

6.729.3.5 virtual int activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

6.729.3.6 virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 712).

```
6.729.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h

6.730 activemq::cmsutil::SessionPool Class Reference

A pool of CMS sessions from the same connection and with the same acknowledge mode.

```
#include <src/main/activemq/cmsutil/SessionPool.h>
```

Public Member Functions

- **SessionPool** (**cms::Connection** *connection, **cms::Session::AcknowledgeMode** acknowledgeMode, **ResourceLifecycleManager** *resourceLifecycleManager)
Constructs a session pool.

- virtual `~SessionPool ()`
Destroys the pooled session objects, but not the underlying session resources.
- virtual `PooledSession * takeSession () throw (cms::CMSEException)`
Takes a session from the pool, creating one if necessary.
- virtual void `returnSession (PooledSession *session)`
Returns a session to the pool.
- `ResourceLifecycleManager * getResourceLifecycleManager ()`

Protected Member Functions

- `SessionPool (const SessionPool &)`
- `SessionPool & operator= (const SessionPool &)`

6.730.1 Detailed Description

A pool of CMS sessions from the same connection and with the same acknowledge mode. Internal session resources are managed through a provided `ResourceLifecycleManager` (p. 3074), not by this pool. This class is thread-safe.

6.730.2 Constructor & Destructor Documentation

6.730.2.1 `activemq::cmsutil::SessionPool::SessionPool (const SessionPool &)`
[inline, protected]

6.730.2.2 `activemq::cmsutil::SessionPool::SessionPool (cms::Connection * connection, cms::Session::AcknowledgeMode ackMode, ResourceLifecycleManager * resourceLifecycleManager)`

Constructs a session pool.

Parameters

connection the connection to be used for creating all sessions.

ackMode the acknowledge mode to be used for all sessions

resourceLifecycleManager the object responsible for managing the lifecycle of any allocated `cms::Session` (p. 3148) resources.

6.730.2.3 `virtual activemq::cmsutil::SessionPool::~SessionPool ()` [virtual]

Destroys the pooled session objects, but not the underlying session resources.

That is the job of the `ResourceLifecycleManager` (p. 3074).

6.730.3 Member Function Documentation

6.730.3.1 `ResourceLifecycleManager* activemq::cmsutil::SessionPool::getResourceLifecycleManager ()` [inline]

6.730.3.2 `SessionPool& activemq::cmsutil::SessionPool::operator= (const SessionPool &)` [inline, protected]

6.730.3.3 `virtual void activemq::cmsutil::SessionPool::returnSession (PooledSession * session)` [virtual]

Returns a session to the pool.

Parameters

session the session to be returned.

6.730.3.4 `virtual PooledSession* activemq::cmsutil::SessionPool::takeSession ()`
`throw (cms::CMSEException)` [virtual]

Takes a session from the pool, creating one if necessary.

Returns

the pooled session object

Exceptions

cms::CMSEException (p. 1074) if an error occurred

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/SessionPool.h`

6.731 activemq::state::SessionState Class Reference

```
#include <src/main/activemq/state/SessionState.h>
```

Public Member Functions

- `SessionState (const Pointer< SessionInfo > &info)`
- `virtual ~SessionState ()`
- `std::string toString () const`
- `const Pointer< SessionInfo > getInfo () const`
- `void addProducer (const Pointer< ProducerInfo > &info)`
- `Pointer< ProducerState > removeProducer (const Pointer< ProducerId > &id)`
- `void addConsumer (const Pointer< ConsumerInfo > &info)`
- `Pointer< ConsumerState > removeConsumer (const Pointer< ConsumerId > &id)`
- `std::vector< Pointer< ProducerState > > getProducerStates () const`

- **Pointer< ProducerState > getProducerState** (const **Pointer< ProducerId >** &id)

- **std::vector< Pointer< ConsumerState > > getConsumerStates** () const

- **Pointer< ConsumerState > getConsumerState** (const **Pointer< ConsumerId >** &id)

- **void checkShutdown** () const

- **void shutdown** ()

6.731.1 Constructor & Destructor Documentation

6.731.1.1 `activemq::state::SessionState::SessionState (const Pointer< SessionInfo > & info)`

6.731.1.2 `virtual activemq::state::SessionState::~~SessionState () [virtual]`

6.731.2 Member Function Documentation

6.731.2.1 `void activemq::state::SessionState::addConsumer (const Pointer< ConsumerInfo > & info) [inline]`

6.731.2.2 `void activemq::state::SessionState::addProducer (const Pointer< ProducerInfo > & info)`

6.731.2.3 `void activemq::state::SessionState::checkShutdown () const`

6.731.2.4 `Pointer<ConsumerState> activemq::state::SessionState::getConsumerState (const Pointer< ConsumerId > & id) [inline]`

6.731.2.5 `std::vector< Pointer<ConsumerState> > activemq::state::SessionState::getConsumerStates () const [inline]`

6.731.2.6 `const Pointer<SessionInfo> activemq::state::SessionState::getInfo () const [inline]`

6.731.2.7 `Pointer<ProducerState> activemq::state::SessionState::getProducerState (const Pointer< ProducerId > & id) [inline]`

6.731.2.8 `std::vector< Pointer<ProducerState> > activemq::state::SessionState::getProducerStates () const [inline]`

6.731.2.9 `Pointer<ConsumerState> activemq::state::SessionState::removeConsumer (const Pointer< ConsumerId > & id) [inline]`

6.731.2.10 `Pointer<ProducerState> activemq::state::SessionState::removeProducer (const Pointer< ProducerId > & id)`

6.731.2.11 `void activemq::state::SessionState::shutdown () [inline]`

6.731.2.12 `std::string activemq::state::SessionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/SessionState.h`

6.732 decaf::util::Set< E > Class Template Reference

A collection that contains no duplicate elements.

```
#include <src/main/decaf/util/Set.h>
```

Inheritance diagram for decaf::util::Set< E >:

Public Member Functions

- virtual `~Set()`

6.732.1 Detailed Description

```
template<typename E> class decaf::util::Set< E >
```

A collection that contains no duplicate elements. More formally, sets contain no pair of elements `e1` and `e2` such that `e1 == e2`, and at most one null element. As implied by its name, this interface models the mathematical set abstraction.

The additional stipulation on constructors is, not surprisingly, that all constructors must create a set that contains no duplicate elements (as defined above).

Note: Great care must be exercised if mutable objects are used as set elements. The behavior of a set is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is an element in the set.

Since

1.0

6.732.2 Constructor & Destructor Documentation

6.732.2.1 `template<typename E> virtual decaf::util::Set< E >::~~Set ()`
[inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Set.h`

6.733 decaf::lang::Short Class Reference

```
#include <src/main/decaf/lang/Short.h>
```

Inheritance diagram for decaf::lang::Short:

Public Member Functions

- **Short** (short value)
- **Short** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Short** ()
- virtual int **compareTo** (const **Short** &s) const
*Compares this **Short** (p. 3220) instance with another.*
- bool **equals** (const **Short** &s) const
- virtual bool **operator==** (const **Short** &s) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Short** &s) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const short &s) const
*Compares this **Short** (p. 3220) instance with another.*
- bool **equals** (const short &s) const
- virtual bool **operator==** (const short &s) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const short &s) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static std::string **toString** (short value)
- static **Short decode** (const std::string &value) throw (exceptions::NumberFormatException)
*Decodes a **String** (p. 3427) into a **Short** (p. 3220).*
- static short **reverseBytes** (short value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.
- static short **parseShort** (const std::string &s, int radix) throw (exceptions::NumberFormatException)
Parses the string argument as a signed short in the radix specified by the second argument.
- static short **parseShort** (const std::string &s) throw (exceptions::NumberFormatException)
Parses the string argument as a signed decimal short.
- static **Short valueOf** (short value)
*Returns a **Short** (p. 3220) instance representing the specified short value.*
- static **Short valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Short** (p. 3220) object holding the value given by the specified std::string.*
- static **Short valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Short** (p. 3220) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const int **SIZE** = 16
Size of this objects primitive type in bits.
- static const short **MAX_VALUE** = (short)0x7FFF
Max Value for this Object's primitive type.
- static const short **MIN_VALUE** = (short)0x8000
Max Value for this Object's primitive type.

6.733.1 Constructor & Destructor Documentation

6.733.1.1 decaf::lang::Short::Short (short value)

Parameters

value - short to wrap

6.733.1.2 `decaf::lang::Short::Short (const std::string & value) throw (exceptions::NumberFormatException)`

Parameters

value - string value to convert to short and wrap

Exceptions

NumberFormatException

6.733.1.3 `virtual decaf::lang::Short::~~Short () [inline, virtual]`

6.733.2 Member Function Documentation

6.733.2.1 `virtual unsigned char decaf::lang::Short::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

6.733.2.2 `virtual int decaf::lang::Short::compareTo (const short & s) const [virtual]`

Compares this **Short** (p. 3220) instance with another.

Parameters

s - the **Short** (p. 3220) instance to be compared

Returns

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< short >` (p. 1126).

6.733.2.3 `virtual int decaf::lang::Short::compareTo (const Short & s) const [virtual]`

Compares this **Short** (p. 3220) instance with another.

Parameters

s - the **Short** (p. 3220) instance to be compared

Returns

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.733.2.4 `static Short decaf::lang::Short::decode (const std::string & value)
throw (exceptions::NumberFormatException) [static]`

Decodes a **String** (p. 3427) into a **Short** (p. 3220).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Short.parseShort** (p. 3226) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified **String** (p. 3427) is the minus sign. No whitespace characters are permitted in the string.

Parameters

value - The string to decode

Returns

a **Short** (p. 3220) object containing the decoded value

Exceptions

NumberFormatException if the string is not formatted correctly.

6.733.2.5 `virtual double decaf::lang::Short::doubleValue () const [inline,
virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.733.2.6 `bool decaf::lang::Short::equals (const Short & s) const [inline]`

Returns

true if the two **Short** (p. 3220) Objects have the same value.

6.733.2.7 `bool decaf::lang::Short::equals (const short & s) const [inline,
virtual]`

Returns

true if the two **Short** (p. 3220) Objects have the same value.

Implements **decaf::lang::Comparable< short >** (p. 1126).

6.733.2.8 `virtual float decaf::lang::Short::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.733.2.9 `virtual int decaf::lang::Short::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.733.2.10 `virtual long long decaf::lang::Short::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.733.2.11 `virtual bool decaf::lang::Short::operator< (const Short & s) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

s - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.733.2.12 `virtual bool decaf::lang::Short::operator< (const short & s) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

s - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< short >` (p.1127).

6.733.2.13 `virtual bool decaf::lang::Short::operator==(const Short & s) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

s - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.733.2.14 `virtual bool decaf::lang::Short::operator==(const short & s) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

s - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< short >` (p.1127).

6.733.2.15 `static short decaf::lang::Short::parseShort (const std::string & s, int radix) throw (exceptions::NumberFormatException)` [static]

Parses the string argument as a signed short in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether `Character.digit(char, int)` (p.1022) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type `NumberFormatException` is thrown if any of the following situations occurs:
* The first argument is null or is a string of length zero. * The radix is either smaller than `Character.MIN_RADIX` (p.1026) or larger than `Character.MAX_RADIX` (p.1026). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type short.

Parameters

s - the **String** (p.3427) containing the short representation to be parsed

radix - the radix to be used while parsing *s*

Returns

the short represented by the string argument in the specified radix.

Exceptions

NumberFormatException - If **String** (p.3427) does not contain a parsable short.

6.733.2.16 `static short decaf::lang::Short::parseShort (const std::string & s)
throw (exceptions::NumberFormatException) [static]`

Parses the string argument as a signed decimal short.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting short value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseShort(const std::string, int)` method.

Parameters

s - **String** (p. 3427) to convert to a short

Returns

the converted short value

Exceptions

NumberFormatException if the string is not a short.

6.733.2.17 `static short decaf::lang::Short::reverseBytes (short value) [static]`

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.

Parameters

value - the short whose bytes we are to reverse

Returns

the reversed short.

6.733.2.18 `virtual short decaf::lang::Short::shortValue () const [inline,
virtual]`

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

6.733.2.19 `static std::string decaf::lang::Short::toString (short value) [static]`

Returns

a string representing the primitive value as Base 10

6.733.2.20 `std::string decaf::lang::Short::toString () const`**Returns**

this **Short** (p. 3220) Object as a **String** (p. 3427) Representation

6.733.2.21 `static Short decaf::lang::Short::valueOf (short value) [static]`

Returns a **Short** (p. 3220) instance representing the specified short value.

Parameters

value - the short to wrap

Returns

the new **Short** (p. 3220) object wrapping value.

6.733.2.22 `static Short decaf::lang::Short::valueOf (const std::string & value)
throw (exceptions::NumberFormatException) [static]`

Returns a **Short** (p. 3220) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal short, exactly as if the argument were given to the `parseShort(std::string)` method. The result is a **Short** (p. 3220) object that represents the short value specified by the string.

Parameters

value - std::string to parse as base 10

Returns

new **Short** (p. 3220) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a decimal short.

6.733.2.23 `static Short decaf::lang::Short::valueOf (const std::string & value, int
radix) throw (exceptions::NumberFormatException) [static]`

Returns a **Short** (p. 3220) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed short in the radix specified by the second argument, exactly as if the argument were given to the `parseShort(std::string, int)` method. The result is a **Short** (p. 3220) object that represents the short value specified by the string.

Parameters

value - std::string to parse as base (radix)

radix - base of the string to parse.

Returns

new **Short** (p. 3220) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a valid short.

6.733.3 Field Documentation

6.733.3.1 `const short decaf::lang::Short::MAX_VALUE = (short)0x7FFF`
[static]

Max Value for this Object's primitive type.

6.733.3.2 `const short decaf::lang::Short::MIN_VALUE = (short)0x8000` [static]

Max Value for this Object's primitive type.

6.733.3.3 `const int decaf::lang::Short::SIZE = 16` [static]

Size of this objects primitive type in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Short.h`

6.734 decaf::internal::nio::ShortArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/ShortArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::ShortArrayBuffer:

Public Member Functions

- **ShortArrayBuffer** (int size, bool readOnly=false) throw (decaf::lang::exceptions::IllegalArgumentException)
*Creates a **ShortArrayBuffer** (p. 3229) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **ShortArrayBuffer** (short *array, int size, int offset, int length, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a **ShortArrayBuffer** (p. 3229) object that wraps the given array.*
- **ShortArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

- **ShortArrayBuffer** (const **ShortArrayBuffer** &other)

Create a **ShortArrayBuffer** (p. 3229) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayAdapter** and when changes are made to that data it is reflected in both.

- virtual ~**ShortArrayBuffer** ()
- virtual short * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 855)

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.

- virtual int **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.
UnsupportedOperationException if the underlying store has no array.

- virtual ShortBuffer * **asReadOnlyBuffer** () const

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the **duplicate** method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only short buffer which the caller then owns.

- virtual ShortBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n + 1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ShortBuffer** (p. 3238).

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual **ShortBuffer** * **duplicate** ()

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new short **Buffer** (p. 855) which the caller owns.

- virtual short **get** () throw (decaf::nio::BufferUnderflowException)

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the short at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return.

- virtual short **get** (int index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

Reads the value at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the short is to be read.

Returns

the short that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or the index is negative.

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual bool **isReadOnly** () const

- virtual **ShortBuffer** & **put** (short value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes the given shorts into this buffer at the current position, and then increments the position.

Parameters

value The shorts value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual ShortBuffer & put (int index, short value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes the given shorts into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The shorts to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

- virtual ShortBuffer * slice () const

Creates a new **ShortBuffer** (p. 3238) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ShortBuffer** (p. 3238) which the caller owns.

Protected Member Functions

- virtual void setReadOnly (bool value)

Sets this **ShortArrayBuffer** (p. 3229) as Read-Only.

6.734.1 Constructor & Destructor Documentation

6.734.1.1 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (int size, bool readOnly = false) throw (decaf::lang::exceptions::IllegalArgumentException)

Creates a **ShortArrayBuffer** (p. 3229) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

size The size of the array, this is the limit we read and write to.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArgumentException if the capacity value is negative.

6.734.1.2 `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (short * array, int size, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a **ShortArrayBuffer** (p. 3229) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array The actual array to wrap.

size The size of the given array.

offset The position that is this buffers start position.

length The limit of how many bytes into the array this Buffer can write.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException if buffer is NULL

IndexOutOfBoundsException if offset is greater than array capacity.

6.734.1.3 `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **ShortArrayBuffer** (p. 3229) will be that of the remaining capacity of the passed buffer.

Parameters

array The ByteArrayAdapter to wrap.

offset The position that is this buffers start position.

length The limit of how many bytes into the array this Buffer can write.

readOnly Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException if array is NULL

IndexOutOfBoundsException if offset + length is greater than array size.

6.734.1.4 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (const ShortArrayBuffer & *other*)

Create a **ShortArrayBuffer** (p. 3229) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

Parameters

other The **ShortArrayBuffer** (p. 3229) this one is to mirror.

6.734.1.5 virtual decaf::internal::nio::ShortArrayBuffer::~~ShortArrayBuffer () [virtual]

6.734.2 Member Function Documentation

6.734.2.1 virtual short* decaf::internal::nio::ShortArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 855)

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 3242).

6.734.2.2 virtual int decaf::internal::nio::ShortArrayBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 3242).

6.734.2.3 `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::asReadOnlyBuffer () const [virtual]`

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only short buffer which the caller then owns.

Implements **decaf::nio::ShortBuffer** (p. 3242).

6.734.2.4 `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ShortBuffer** (p. 3238).

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 3243).

6.734.2.5 `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::duplicate () [virtual]`

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new short **Buffer** (p. 855) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 3243).

6.734.2.6 virtual short decaf::internal::nio::ShortArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the short at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return.

Implements **decaf::nio::ShortBuffer** (p. 3245).

6.734.2.7 virtual short decaf::internal::nio::ShortArrayBuffer::get (int *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the short is to be read.

Returns

the short that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or the index is negative.

Implements **decaf::nio::ShortBuffer** (p. 3244).

6.734.2.8 `virtual bool decaf::internal::nio::ShortArrayBuffer::hasArray () const`
`[inline, virtual]`

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements `decaf::nio::ShortBuffer` (p. 3245).

6.734.2.9 `virtual bool decaf::internal::nio::ShortArrayBuffer::isReadOnly ()`
`const [inline, virtual]`

6.734.2.10 `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put`
`(int index, short value) throw (`
`lang::exceptions::IndexOutOfBoundsException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given shorts into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The shorts to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements `decaf::nio::ShortBuffer` (p. 3246).

6.734.2.11 `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put`
`(short value) throw (decaf::nio::BufferOverflowException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given shorts into this buffer at the current position, and then increments the position.

Parameters

value The shorts value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 3246).

6.734.2.12 `virtual void decaf::internal::nio::ShortArrayBuffer::setReadOnly (bool value)` [inline, protected, virtual]

Sets this **ShortArrayBuffer** (p. 3229) as Read-Only.

Parameters

value Boolean value, true if this buffer is to be read-only, false otherwise.

6.734.2.13 `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::slice () const` [virtual]

Creates a new **ShortBuffer** (p. 3238) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ShortBuffer** (p. 3238) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 3248).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ShortArrayBuffer.h`

6.735 decaf::nio::ShortBuffer Class Reference

This class defines four categories of operations upon short buffers:

```
#include <src/main/decaf/nio/ShortBuffer.h>
```

Inheritance diagram for decaf::nio::ShortBuffer:

Public Member Functions

- virtual `~ShortBuffer ()`
- virtual `std::string toString () const`
- virtual `short * array ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)`
Returns the short array that backs this buffer (optional operation).
- virtual `int arrayOffset ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)`
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual `ShortBuffer * asReadOnlyBuffer () const =0`
Creates a new, read-only short buffer that shares this buffer's content.
- virtual `ShortBuffer & compact ()=0 throw (ReadOnlyBufferException)`
Compacts this buffer.
- virtual `ShortBuffer * duplicate ()=0`
Creates a new short buffer that shares this buffer's content.
- virtual `short get ()=0 throw (BufferUnderflowException)`
Relative get method.
- virtual `short get (int index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
Absolute get method.
- `ShortBuffer & get (std::vector< short > buffer) throw (BufferUnderflowException)`
Relative bulk get method.
- `ShortBuffer & get (short *buffer, int size, int offset, int length) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)`
Relative bulk get method.
- virtual `bool hasArray () const =0`
Tells whether or not this buffer is backed by an accessible short array.
- `ShortBuffer & put (ShortBuffer &src) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException)`
This method transfers the shorts remaining in the given source buffer into this buffer.
- `ShortBuffer & put (const short *buffer, int size, int offset, int length) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)`
This method transfers shorts into this buffer from the given source array.

- **ShortBuffer** & **put** (std::vector< short > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source shorts array into this buffer.
- virtual **ShortBuffer** & **put** (short value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given shorts into this buffer at the current position, and then increments the position.
- virtual **ShortBuffer** & **put** (int index, short value)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given shorts into this buffer at the given index.
- virtual **ShortBuffer** * **slice** () const =0
*Creates a new **ShortBuffer** (p. 3238) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **ShortBuffer** &value) const
- virtual bool **equals** (const **ShortBuffer** &value) const
- virtual bool **operator==** (const **ShortBuffer** &value) const
- virtual bool **operator<** (const **ShortBuffer** &value) const

Static Public Member Functions

- static **ShortBuffer** * **allocate** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)
Allocates a new Double buffer.
- static **ShortBuffer** * **wrap** (short *array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
*Wraps the passed buffer with a new **ShortBuffer** (p. 3238).*
- static **ShortBuffer** * **wrap** (std::vector< short > &buffer)
*Wraps the passed STL short Vector in a **ShortBuffer** (p. 3238).*

Protected Member Functions

- **ShortBuffer** (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException)
*Creates a **ShortBuffer** (p. 3238) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.735.1 Detailed Description

This class defines four categories of operations upon short buffers: o Absolute and relative get and put methods that read and write single shorts; o Relative bulk get methods that transfer contiguous sequences of shorts from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of shorts from a short array or some other short buffer into this buffer o Methods for compacting, duplicating, and slicing a short buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing short array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.735.2 Constructor & Destructor Documentation

6.735.2.1 `decaf::nio::ShortBuffer::ShortBuffer (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException) [protected]`

Creates a **ShortBuffer** (p. 3238) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity The size and limit of the **Buffer** (p. 855) in doubles

Exceptions

IllegalArgumentException if capacity is negative.

6.735.2.2 `virtual decaf::nio::ShortBuffer::~~ShortBuffer () [inline, virtual]`

6.735.3 Member Function Documentation

6.735.3.1 `static ShortBuffer* decaf::nio::ShortBuffer::allocate (int capacity) throw (decaf::lang::exceptions::IllegalArgumentException) [static]`

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity The size of the Double buffer in shorts.

Returns

the **ShortBuffer** (p. 3238) that was allocated, caller owns.

6.735.3.2 `virtual short* decaf::nio::ShortBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 855)

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3234).

6.735.3.3 `virtual int decaf::nio::ShortBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2966) if this **Buffer** (p. 855) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3234).

6.735.3.4 `virtual ShortBuffer* decaf::nio::ShortBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only short buffer which the caller then owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3235).

6.735.3.5 **virtual ShortBuffer& decaf::nio::ShortBuffer::compact () throw (ReadOnlyBufferException)** [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 860) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 859) - 1 is copied to index $n = \text{limit}()$ (p. 859) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ShortBuffer** (p. 3238).

Exceptions

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3235).

6.735.3.6 **virtual int decaf::nio::ShortBuffer::compareTo (const ShortBuffer & value) const** [virtual]

6.735.3.7 **virtual ShortBuffer* decaf::nio::ShortBuffer::duplicate ()** [pure virtual]

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new short **Buffer** (p. 855) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3235).

6.735.3.8 virtual bool decaf::nio::ShortBuffer::equals (const ShortBuffer & *value*) const [virtual]

6.735.3.9 ShortBuffer& decaf::nio::ShortBuffer::get (std::vector< short > *buffer*) throw (BufferUnderflowException)

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than length shorts remaining in this buffer.

6.735.3.10 virtual short decaf::nio::ShortBuffer::get (int *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index The index in the **Buffer** (p. 855) where the short is to be read.

Returns

the short that is located at the given index.

Exceptions

IndexOutOfBoundsException if index is not smaller than the buffer's limit, or the index is negative.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3236).

6.735.3.11 ShortBuffer& decaf::nio::ShortBuffer::get (short * *buffer*, int *size*, int *offset*, int *length*) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

Relative bulk get method.

This method transfers shorts from this buffer into the given destination array. If there are fewer shorts remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 860), then no bytes are transferred and a **BufferUnderflowException** (p. 882) is thrown.

Otherwise, this method copies length shorts from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters

buffer The pointer to an allocated buffer to fill.
size The size of the buffer provided.
offset The position in the buffer to start filling.
length The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 855).

Exceptions

BufferUnderflowException (p. 882) if there are fewer than length shorts remaining in this buffer
NullPointerException if the passed buffer is null.
IndexOutOfBoundsException if the preconditions of size, offset, or length are not met.

6.735.3.12 `virtual short decaf::nio::ShortBuffer::get () throw (BufferUnderflowException) [pure virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the short at the current position.

Exceptions

BufferUnderflowException (p. 882) if there no more data to return.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3236).

6.735.3.13 `virtual bool decaf::nio::ShortBuffer::hasArray () const [pure virtual]`

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3237).

- 6.735.3.14** `virtual bool decaf::nio::ShortBuffer::operator< (const ShortBuffer & value) const` [virtual]
- 6.735.3.15** `virtual bool decaf::nio::ShortBuffer::operator== (const ShortBuffer & value) const` [virtual]
- 6.735.3.16** `virtual ShortBuffer& decaf::nio::ShortBuffer::put (short value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given shorts into this buffer at the current position, and then increments the position.

Parameters

value The shorts value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if this buffer's current position is not smaller than its limit.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p. 3237).

- 6.735.3.17** `virtual ShortBuffer& decaf::nio::ShortBuffer::put (int index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given shorts into this buffer at the given index.

Parameters

index The position in the **Buffer** (p. 855) to write the data.

value The shorts to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException if index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p. 3237).

- 6.735.3.18** `ShortBuffer& decaf::nio::ShortBuffer::put (const short * buffer, int size, int offset, int length) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)`

This method transfers shorts into this buffer from the given source array.

If there are more shorts to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 860), then no shorts are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

- buffer* The array from which shorts are to be read.
- size* The size of the buffer passed.
- offset* The offset within the array of the first char to be read.
- length* The number of shorts to be read from the given array.

Returns

a reference to this buffer.

Exceptions

- BufferOverflowException** (p. 880) if there is insufficient space in this buffer
- ReadOnlyBufferException** (p. 2966) if this buffer is read-only
- NullPointerException** if the passed buffer is null.
- IndexOutOfBoundsException** if the preconditions of size, offset, or length are not met.

6.735.3.19 ShortBuffer& decaf::nio::ShortBuffer::put (std::vector< short > & buffer) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source shorts array into this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size()`.

Parameters

- buffer* The buffer whose contents are copied to this **ShortBuffer** (p. 3238).

Returns

a reference to this buffer.

Exceptions

- BufferOverflowException** (p. 880) if there is insufficient space in this buffer.
- ReadOnlyBufferException** (p. 2966) if this buffer is read-only.

6.735.3.20 ShortBuffer& decaf::nio::ShortBuffer::put (ShortBuffer & src) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException)

This method transfers the shorts remaining in the given source buffer into this buffer.

If there are more shorts remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 860), then no shorts are transferred and a **BufferOverflowException** (p. 880) is thrown.

Otherwise, this method copies `n = src.remaining()` shorts from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

src The buffer to take shorts from an place in this one.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 880) if there is insufficient space in this buffer for the remaining shorts in the source buffer.

IllegalArgumentException if the source buffer is this buffer.

ReadOnlyBufferException (p. 2966) if this buffer is read-only.

6.735.3.21 virtual ShortBuffer* decaf::nio::ShortBuffer::slice () const [pure virtual]

Creates a new **ShortBuffer** (p. 3238) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ShortBuffer** (p. 3238) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3238).

6.735.3.22 virtual std::string decaf::nio::ShortBuffer::toString () const [virtual]

Returns

a `std::string` describing this object

6.735.3.23 static ShortBuffer* decaf::nio::ShortBuffer::wrap (short * array, int size, int offset, int length) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [static]

Wraps the passed buffer with a new **ShortBuffer** (p. 3238).

The new buffer will be backed by the given short array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

- array*** The array that will back the new buffer.
- size*** The size of the passed in array.
- offset*** The offset of the subarray to be used.
- length*** The length of the subarray to be used.

Returns

a new **ShortBuffer** (p. 3238) that is backed by `buffer`, caller owns.

Exceptions

- NullPointerException*** if the array pointer is `NULL`.
- IndexOutOfBoundsException*** if the preconditions of `size`, `offset`, or `length` are not met.

6.735.3.24 `static ShortBuffer* decaf::nio::ShortBuffer::wrap (std::vector< short > & buffer) [static]`

Wraps the passed STL short Vector in a **ShortBuffer** (p. 3238).

The new buffer will be backed by the given short array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

- buffer*** The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new **ShortBuffer** (p. 3238) that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/ShortBuffer.h`

6.736 activemq::commands::ShutdownInfo Class Reference

```
#include <src/main/activemq/commands/ShutdownInfo.h>
```

Inheritance diagram for `activemq::commands::ShutdownInfo`:

Public Member Functions

- **ShutdownInfo** ()
- virtual **~ShutdownInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ShutdownInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual bool **isShutdownInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_SHUTDOWNINFO** = 11

6.736.1 Constructor & Destructor Documentation

6.736.1.1 **activemq::commands::ShutdownInfo::ShutdownInfo** ()

6.736.1.2 **virtual activemq::commands::ShutdownInfo::~~ShutdownInfo** ()
[virtual]

6.736.2 Member Function Documentation

6.736.2.1 **virtual ShutdownInfo* activemq::commands::ShutdownInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1554).

6.736.2.2 `virtual void activemq::commands::ShutdownInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.736.2.3 `virtual bool activemq::commands::ShutdownInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.736.2.4 `virtual unsigned char activemq::commands::ShutdownInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1553) type copy.

Implements `activemq::commands::DataStructure` (p. 1557).

6.736.2.5 `virtual bool activemq::commands::ShutdownInfo::isShutdownInfo () const [inline, virtual]`

Returns

an answer of true to the `isShutdownInfo()` (p. 3251) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 700).

6.736.2.6 `virtual std::string activemq::commands::ShutdownInfo::toString () const [virtual]`

Returns a string containing the information for this `DataStructure` (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 700).

6.736.2.7 `virtual Pointer<Command> activemq::commands::ShutdownInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.736.3 Field Documentation

6.736.3.1 `const unsigned char activemq::commands::ShutdownInfo::ID_ - SHUTDOWNINFO = 11` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ShutdownInfo.h`

6.737 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3252).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller`:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.737.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3252). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.737.2 Constructor & Destructor Documentation

6.737.2.1 **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::ShutdownInfoMarshaller** () [inline]

6.737.2.2 virtual **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller** () [inline, virtual]

6.737.3 Member Function Documentation

6.737.3.1 virtual **commands::DataStructure*** **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.737.3.2 virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::getDataStructureType
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.737.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::looseMarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * *dataOut*) throw (
decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**
(p. 729).

6.737.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::looseUnmarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**
(p. 730).

6.737.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

6.737.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 732).

6.737.3.7 `virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 733).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h`

6.738 `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ShutdownInfoMarshaller` (p. 3256).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller`:

Public Member Functions

- `ShutdownInfoMarshaller ()`
- `virtual ~ShutdownInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.738.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3256). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.738.2 Constructor & Destructor Documentation

6.738.2.1 **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::ShutdownInfoMar**
() [inline]

6.738.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::~~ShutdownInfoMa
() [inline, virtual]

6.738.3 Member Function Documentation

6.738.3.1 **virtual commands::DataStructure* ac-**
tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.738.3.2 **virtual unsigned char ac-**
tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::getDataStructureTy
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.738.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.738.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

6.738.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

```
6.738.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

```
6.738.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h`

6.739 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3260).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller**:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.739.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3260). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.739.2 Constructor & Destructor Documentation

6.739.2.1 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::ShutdownInfoMarshaller () [inline]

6.739.2.2 virtual
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller () [inline, virtual]

6.739.3 Member Function Documentation

6.739.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.739.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.739.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 716).

6.739.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 717).

6.739.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

```
6.739.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 719).

```
6.739.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h`

6.740 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3264).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller**:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.740.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3264). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.740.2 Constructor & Destructor Documentation

6.740.2.1 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::ShutdownInfoMarshaller () [inline]

6.740.2.2 virtual
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller () [inline, virtual]

6.740.3 Member Function Documentation

6.740.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.740.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.740.3.3 virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 722).

6.740.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 723).

6.740.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```
6.740.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 726).

```
6.740.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h`

6.741 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3268).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller**:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.741.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3268). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.741.2 Constructor & Destructor Documentation

6.741.2.1 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::ShutdownInfoMarshaller () [inline]

6.741.2.2 virtual
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller () [inline, virtual]

6.741.3 Member Function Documentation

6.741.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.741.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.741.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 702).

6.741.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 703).

6.741.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

```
6.741.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.741.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h`

6.742 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3272).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller**:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.742.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3272). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.742.2 Constructor & Destructor Documentation

6.742.2.1 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::ShutdownInfoMarshaller () [inline]

6.742.2.2 virtual
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller () [inline, virtual]

6.742.3 Member Function Documentation

6.742.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.742.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.742.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 709).

6.742.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 710).

6.742.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

```
6.742.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 712).

```
6.742.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h`

6.743 decaf::security::SignatureException Class Reference

```
#include <src/main/decaf/security/SignatureException.h>
```

Inheritance diagram for decaf::security::SignatureException:

Public Member Functions

- **SignatureException** () throw ()
Default Constructor.
- **SignatureException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **SignatureException** (const **SignatureException** &ex) throw ()
Copy Constructor.
- **SignatureException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SignatureException** (const std::exception *cause) throw ()
Constructor.
- **SignatureException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **SignatureException** * **clone** () const
Clones this exception.
- virtual ~**SignatureException** () throw ()

6.743.1 Constructor & Destructor Documentation

6.743.1.1 decaf::security::SignatureException::SignatureException () throw () [inline]

Default Constructor.

6.743.1.2 `decaf::security::SignatureException::SignatureException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.743.1.3 `decaf::security::SignatureException::SignatureException (const SignatureException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.743.1.4 `decaf::security::SignatureException::SignatureException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.743.1.5 `decaf::security::SignatureException::SignatureException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.743.1.6 `decaf::security::SignatureException::SignatureException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

- file* name where exception occurs
- lineNumber* line number where the exception occurred.
- msg* message to report
- ... list of primitives that are formatted into the message

6.743.1.7 virtual decaf::security::SignatureException::~SignatureException ()
throw () [inline, virtual]

6.743.2 Member Function Documentation

6.743.2.1 virtual SignatureException* decaf::security::SignatureException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1847).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SignatureException.h**

6.744 decaf::util::logging::SimpleFormatter Class Reference

Print a brief summary of the **LogRecord** (p. 2261) in a human readable format.

```
#include <src/main/decaf/util/logging/SimpleFormatter.h>
```

Inheritance diagram for decaf::util::logging::SimpleFormatter:

Public Member Functions

- **SimpleFormatter** ()
- virtual ~**SimpleFormatter** ()
- virtual std::string **format** (const **LogRecord** &record) const
Format the given log record and return the formatted string.

6.744.1 Detailed Description

Print a brief summary of the **LogRecord** (p. 2261) in a human readable format. The summary will typically be 1 or 2 lines.

Since

1.0

6.744.2 Constructor & Destructor Documentation

6.744.2.1 `decaf::util::logging::SimpleFormatter::SimpleFormatter ()`

6.744.2.2 `virtual decaf::util::logging::SimpleFormatter::~~SimpleFormatter ()`
[virtual]

6.744.3 Member Function Documentation

6.744.3.1 `virtual std::string decaf::util::logging::SimpleFormatter::format (const LogRecord & record) const` [virtual]

Format the given log record and return the formatted string.

Parameters

record The Log Record to Format.

Implements **decaf::util::logging::Formatter** (p. 1839).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/SimpleFormatter.h`

6.745 decaf::util::logging::SimpleLogger Class Reference

```
#include <src/main/decaf/util/logging/SimpleLogger.h>
```

Public Member Functions

- **SimpleLogger** (const std::string &name)
Constructor.
- virtual **~SimpleLogger** ()
Destructor.
- virtual void **mark** (const std::string &message)
*Log a Mark Block **Level** (p. 2185) Log.*
- virtual void **debug** (const std::string &file, const int line, const std::string &message)
*Log a Debug **Level** (p. 2185) Log.*

- virtual void **info** (const std::string &file, const int line, const std::string &message)
*Log a Informational **Level** (p. 2185) Log.*
- virtual void **warn** (const std::string &file, const int line, const std::string &message)
*Log a Warning **Level** (p. 2185) Log.*
- virtual void **error** (const std::string &file, const int line, const std::string &message)
*Log a Error **Level** (p. 2185) Log.*
- virtual void **fatal** (const std::string &file, const int line, const std::string &message)
*Log a Fatal **Level** (p. 2185) Log.*
- virtual void **log** (const std::string &message)
No-frills log.

6.745.1 Constructor & Destructor Documentation

6.745.1.1 decaf::util::logging::SimpleLogger::SimpleLogger (const std::string &name)

Constructor.

6.745.1.2 virtual decaf::util::logging::SimpleLogger::~SimpleLogger () [virtual]

Destructor.

6.745.2 Member Function Documentation

6.745.2.1 virtual void decaf::util::logging::SimpleLogger::debug (const std::string &file, const int line, const std::string &message) [virtual]

Log a Debug **Level** (p. 2185) Log.

6.745.2.2 virtual void decaf::util::logging::SimpleLogger::error (const std::string &file, const int line, const std::string &message) [virtual]

Log a Error **Level** (p. 2185) Log.

6.745.2.3 virtual void decaf::util::logging::SimpleLogger::fatal (const std::string &file, const int line, const std::string &message) [virtual]

Log a Fatal **Level** (p. 2185) Log.

6.745.2.4 virtual void decaf::util::logging::SimpleLogger::info (const std::string &file, const int line, const std::string &message) [virtual]

Log a Informational **Level** (p. 2185) Log.

6.745.2.5 virtual void decaf::util::logging::SimpleLogger::log (const std::string & *message*) [virtual]

No-frills log.

6.745.2.6 virtual void decaf::util::logging::SimpleLogger::mark (const std::string & *message*) [virtual]

Log a Mark Block **Level** (p. 2185) Log.

6.745.2.7 virtual void decaf::util::logging::SimpleLogger::warn (const std::string & *file*, const int *line*, const std::string & *message*) [virtual]

Log a Warning **Level** (p. 2185) Log.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**SimpleLogger.h**

6.746 decaf::net::Socket Class Reference

```
#include <src/main/decaf/net/Socket.h>
```

Inheritance diagram for decaf::net::Socket:

Public Member Functions

- **Socket** ()
*Creates an unconnected **Socket** (p. 3281) using the set **SocketImplFactory** (p. 3314) or if non is set than the default **SocketImpl** type is created.*
- **Socket** (**SocketImpl** *impl)
*Creates a **Socket** (p. 3281) wrapping the provided **SocketImpl** (p. 3306) instance, this **Socket** (p. 3281) is considered unconnected.*
- **Socket** (const **InetAddress** *address, int port)
*Creates a new **Socket** (p. 3281) instance and connects it to the given address and port.*
- **Socket** (const **InetAddress** *address, int port, const **InetAddress** *localAddress, int localPort)
*Creates a new **Socket** (p. 3281) instance and connects it to the given address and port.*
- **Socket** (const std::string &host, int port)
*Creates a new **Socket** (p. 3281) instance and connects it to the given host and port.*
- **Socket** (const std::string &host, int port, const **InetAddress** *localAddress, int localPort)
*Creates a new **Socket** (p. 3281) instance and connects it to the given host and port.*

- virtual `~Socket ()`
- virtual void **bind** (const std::string &ipaddress, int port) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException)
*Binds this **Socket** (p. 3281) to the given local address and port.*
- virtual void **close** () throw (decaf::io::IOException)
*Closes the **Socket** (p. 3281).*
- virtual void **connect** (const std::string &host, int port) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException)
Connects to the specified destination.
- virtual void **connect** (const std::string &host, int port, int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException)
Connects to the specified destination, with a specified timeout value.
- bool **isConnected** () const
Indicates whether or not this socket is connected to an end point.
- bool **isClosed** () const
- bool **isBound** () const
- bool **isInputShutdown** () const
- bool **isOutputShutdown** () const
- virtual `decaf::io::InputStream * getInputStream ()` throw (decaf::io::IOException)
Gets the `InputStream` for this socket if its connected.
- virtual `decaf::io::OutputStream * getOutputStream ()` throw (decaf::io::IOException)
Gets the `OutputStream` for this socket if it is connected.
- int **getPort** () const
*Gets the on the remote host this **Socket** (p. 3281) is connected to.*
- int **getLocalPort** () const
Gets the local port the socket is bound to.
- std::string **getInetAddress** () const
Returns the address to which the socket is connected.
- std::string **getLocalAddress** () const
Gets the local address to which the socket is bound.
- virtual void **shutdownInput** () throw (decaf::io::IOException)
Shuts down the `InputStream` for this socket essentially marking it as EOF.
- virtual void **shutdownOutput** () throw (decaf::io::IOException)
Shuts down the `OutputStream` for this socket, any data already written to the socket will be sent, any further calls to `OutputStream::write` will throw an `IOException`.
- virtual int **getSoLinger** () const throw (SocketException)

Gets the linger time for the socket, `SO_LINGER`.

- virtual void **setSoLinger** (bool state, int timeout) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException)
Sets the linger time (`SO_LINGER`) using a specified time value, this limits of this value are platform specific.
- virtual bool **getKeepAlive** () const throw (SocketException)
Gets the keep alive flag for this socket, `SO_KEEPAIVE`.
- virtual void **setKeepAlive** (bool keepAlive) throw (SocketException)
Enables/disables the keep alive flag for this socket, `SO_KEEPAIVE`.
- virtual int **getReceiveBufferSize** () const throw (SocketException)
Gets the receive buffer size for this socket, `SO_RCVBUF`.
- virtual void **setReceiveBufferSize** (int size) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException)
Sets the receive buffer size for this socket, `SO_RCVBUF`.
- virtual bool **getReuseAddress** () const throw (SocketException)
Gets the reuse address flag, `SO_REUSEADDR`.
- virtual void **setReuseAddress** (bool reuse) throw (SocketException)
Sets the reuse address flag, `SO_REUSEADDR`.
- virtual int **getSendBufferSize** () const throw (SocketException)
Gets the send buffer size for this socket, `SO_SNDBUF`, this value is used by the platform socket to buffer data written to the socket.
- virtual void **setSendBufferSize** (int size) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException)
Gets the send buffer size for this socket, `SO_SNDBUF`, this value is used by the platform socket to buffer data written to the socket.
- virtual int **getSoTimeout** () const throw (SocketException)
Gets the timeout for socket operations, `SO_TIMEOUT`.
- virtual void **setSoTimeout** (int timeout) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException)
Sets the timeout for socket operations, `SO_TIMEOUT`.
- virtual bool **getTcpNoDelay** () const throw (SocketException)
Gets the Status of the `TCP_NODELAY` setting for this socket.
- virtual void **setTcpNoDelay** (bool value) throw (SocketException)
*Sets the Status of the `TCP_NODELAY` param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p.3281).*
- virtual int **getTrafficClass** () const throw (SocketException)

*Gets the Traffic Class setting for this **Socket** (p. 3281), sometimes referred to as Type of Service setting.*

- virtual void **setTrafficClass** (int value) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException)

*Gets the Traffic Class setting for this **Socket** (p. 3281), sometimes referred to as Type of Service setting.*

- virtual bool **getOOBInline** () const throw (SocketException)

Gets the value of the OOBINLINE for this socket.

- virtual void **setOOBInline** (bool value) throw (SocketException)

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

- virtual void **sendUrgentData** (int data) throw (decaf::io::IOException)

*Sends on byte of urgent data to the **Socket** (p. 3281).*

- virtual std::string **toString** () const

Static Public Member Functions

- static void **setSocketImplFactory** (SocketImplFactory *factory) throw (decaf::io::IOException, decaf::net::SocketException)

*Sets the instance of a **SocketImplFactory** (p. 3314) that the **Socket** (p. 3281) class should use when new instances of this class are created.*

Protected Member Functions

- void **accepted** ()
- void **initSocketImpl** (const std::string &address, int port, const InetAddress *localAddress, int localPort) throw (decaf::io::IOException, decaf::net::UnknownHostException)
- void **checkClosed** () const throw (decaf::io::IOException)
- void **ensureCreated** () const throw (decaf::io::IOException)

Protected Attributes

- SocketImpl * impl

Friends

- class ServerSocket

6.746.1 Detailed Description

Since

1.0

6.746.2 Constructor & Destructor Documentation

6.746.2.1 decaf::net::Socket::Socket ()

Creates an unconnected **Socket** (p. 3281) using the set **SocketImplFactory** (p. 3314) or if non is set than the default **SocketImpl** type is created.

6.746.2.2 decaf::net::Socket::Socket (SocketImpl * impl)

Creates a **Socket** (p. 3281) wrapping the provided **SocketImpl** (p. 3306) instance, this **Socket** (p. 3281) is considered unconnected.

The **Socket** (p. 3281) class takes ownership of this **SocketImpl** (p. 3306) pointer and will delete it when the **Socket** (p. 3281) class is destroyed.

Parameters

impl The **SocketImpl** (p. 3306) instance to wrap.

Exceptions

NullPointerException if the passed **SocketImpl** (p. 3306) is Null.

6.746.2.3 decaf::net::Socket::Socket (const InetAddress * address, int port)

Creates a new **Socket** (p. 3281) instance and connects it to the given address and port.

If there is a **SocketImplFactory** (p. 3314) set then the **SocketImpl** is created using the factory otherwise the default **Socket** (p. 3281) implementation is used.

If the host parameter is empty then the loop back address is used.

Parameters

address The address to connect to.

port The port number to connect to [0...65535]

Exceptions

UnknownHostException (p. 3649) if the host cannot be resolved.

IOException if an I/O error occurs while connecting the **Socket** (p. 3281).

NullPointerException if the **InetAddress** (p. 1884) instance is NULL.

IllegalArgumentException if the port is not in range [0...65535]

6.746.2.4 decaf::net::Socket::Socket (const InetAddress * address, int port, const InetAddress * localAddress, int localPort)

Creates a new **Socket** (p. 3281) instance and connects it to the given address and port.

If there is a **SocketImplFactory** (p. 3314) set then the **SocketImpl** is created using the factory otherwise the default **Socket** (p. 3281) implementation is used. The **Socket** (p. 3281) will also bind to the local address and port specified.

Parameters

address The address to connect to.
port The port number to connect to [0...65535]
localAddress The IP address on the local machine to bind to.
localPort The port on the local machine to bind to.

Exceptions

UnknownHostException (p. 3649) if the host cannot be resolved.
IOException if an I/O error occurs while connecting the **Socket** (p. 3281).
NullPointerException if the **InetAddress** (p. 1884) instance is NULL.
IllegalArgumentException if the port is not in range [0...65535]

6.746.2.5 decaf::net::Socket::Socket (const std::string & host, int port)

Creates a new **Socket** (p. 3281) instance and connects it to the given host and port.

If there is a **SocketImplFactory** (p. 3314) set then the **SocketImpl** is created using the factory otherwise the default **Socket** (p. 3281) implementation is used.

If the host parameter is empty then the loop back address is used.

Parameters

host The host name or IP address to connect to, empty string means loopback.
port The port number to connect to [0...65535]

Exceptions

UnknownHostException (p. 3649) if the host cannot be resolved.
IOException if an I/O error occurs while connecting the **Socket** (p. 3281).
IllegalArgumentException if the port is not in range [0...65535]

6.746.2.6 decaf::net::Socket::Socket (const std::string & host, int port, const InetAddress * localAddress, int localPort)

Creates a new **Socket** (p. 3281) instance and connects it to the given host and port.

If there is a **SocketImplFactory** (p. 3314) set then the **SocketImpl** is created using the factory otherwise the default **Socket** (p. 3281) implementation is used.

If the host parameter is empty then the loop back address is used.

Parameters

host The host name or IP address to connect to, empty string means loopback.
port The port number to connect to [0...65535]
localAddress The IP address on the local machine to bind to.
localPort The port on the local machine to bind to.

Exceptions

UnknownHostException (p. 3649) if the host cannot be resolved.

IOException if an I/O error occurs while connecting the **Socket** (p. 3281).

IllegalArgumentOutOfRangeException if the port is not in range [0...65535]

6.746.2.7 virtual `decaf::net::Socket::~~Socket ()` [virtual]

6.746.3 Member Function Documentation

6.746.3.1 void `decaf::net::Socket::accepted ()` [protected]

6.746.3.2 virtual void `decaf::net::Socket::bind (const std::string & ipaddress, int port) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentOutOfRangeException)` [virtual]

Binds this **Socket** (p. 3281) to the given local address and port.

If the **SocketAddress** (p. 3297) value is NULL then the **Socket** (p. 3281) will be bound to an available local address and port.

Parameters

ipaddress The local address and port to bind the socket to.

port The port on the local machine to bind to.

Exceptions

IOException if an error occurs during the bind operation.

IllegalArgumentOutOfRangeException if the **Socket** (p. 3281) can't process the subclass of **SocketAddress** (p. 3297) that has been provided.

6.746.3.3 void `decaf::net::Socket::checkClosed () const throw (decaf::io::IOException)` [protected]

6.746.3.4 virtual void `decaf::net::Socket::close () throw (decaf::io::IOException)` [virtual]

Closes the **Socket** (p. 3281).

Once closed a **Socket** (p. 3281) cannot be connected or otherwise operated upon, a new **Socket** (p. 3281) instance must be created.

Exceptions

IOException if an I/O error occurs while closing the **Socket** (p. 3281).

Implements **decaf::io::Closeable** (p. 1066).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2679).

6.746.3.5 virtual void decaf::net::Socket::connect (const std::string & *host*, int *port*, int *timeout*) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException) [virtual]

Connects to the specified destination, with a specified timeout value.

If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3319) is thrown. A timeout value of zero is treated as an infinite timeout.

Parameters

host The host name or IP address of the remote host to connect to.

port The port on the remote host to connect to.

timeout The number of Milliseconds to wait before treating the connection as failed.

Exceptions

IOException Thrown if a failure occurred in the connect.

SocketTimeoutException (p. 3319) if the timeout for connection is exceeded.

IllegalArgumentException if the timeout value is negative or the endpoint is invalid.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2679).

6.746.3.6 virtual void decaf::net::Socket::connect (const std::string & *host*, int *port*) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException) [virtual]

Connects to the specified destination.

Parameters

host The host name or IP address of the remote host to connect to.

port The port on the remote host to connect to.

Exceptions

IOException Thrown if a failure occurred in the connect.

IllegalArgumentException if the timeout value is negative or the endpoint is invalid.

6.746.3.7 void decaf::net::Socket::ensureCreated () const throw (decaf::io::IOException) [protected]

6.746.3.8 std::string decaf::net::Socket::getInetAddress () const

Returns the address to which the socket is connected.

Returns

the remote IP address to which this socket is connected, or null if the socket is not connected.

6.746.3.9 `virtual decaf::io::InputStream* decaf::net::Socket::getInputStream ()
throw (decaf::io::IOException) [virtual]`

Gets the InputStream for this socket if its connected.

The pointer returned is the property of the associated **Socket** (p. 3281) and should not be deleted by the caller.

When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broken the read calls will normally throw an exception to indicate the failure.

Closing the InputStream will also close the underlying **Socket** (p. 3281).

Returns

The InputStream for this socket.

Exceptions

IOException if an error occurs during creation of the InputStream, also if the **Socket** (p. 3281) is not connected or the input has been shutdown previously.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2680).

6.746.3.10 `virtual bool decaf::net::Socket::getKeepAlive () const throw (
SocketException) [virtual]`

Gets the keep alive flag for this socket, SO_KEEPALIVE.

Returns

true if keep alive is enabled for this socket.

Exceptions

SocketException (p. 3298) if the operation fails.

6.746.3.11 `std::string decaf::net::Socket::getLocalAddress () const`

Gets the local address to which the socket is bound.

Returns

the local address to which the socket is bound or InetAddress.anyLocalAddress() if the socket is not bound yet.

6.746.3.12 `int decaf::net::Socket::getLocalPort () const`

Gets the local port the socket is bound to.

Returns

the local port the socket was bound to, or -1 if the socket is not bound.

6.746.3.13 `virtual bool decaf::net::Socket::getOOBInline () const throw (SocketException) [virtual]`

Gets the value of the OOBINLINE for this socket.

Returns

true if OOBINLINE is enabled, false otherwise.

Exceptions

SocketException (p. 3298) if an error is encountered while performing this operation.

6.746.3.14 `virtual decaf::io::OutputStream* decaf::net::Socket::getOutputStream () throw (decaf::io::IOException) [virtual]`

Gets the OutputStream for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 3281) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 3281) will also close the underlying **Socket** (p. 3281).

Returns

the OutputStream for this socket.

Exceptions

IOException if an error occurs during the creation of this OutputStream, or if the **Socket** (p. 3281) is closed or the output has been shutdown previously.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2681).

6.746.3.15 `int decaf::net::Socket::getPort () const`

Gets the on the remote host this **Socket** (p. 3281) is connected to.

Returns

the port on the remote host the socket is connected to, or 0 if not connected.

6.746.3.16 `virtual int decaf::net::Socket::getReceiveBufferSize () const throw (SocketException) [virtual]`

Gets the receive buffer size for this socket, SO_RCVBUF.

This is the buffer used by the underlying platform socket to buffer received data.

Returns

the receive buffer size in bytes.

Exceptions

SocketException (p. 3298) if the operation fails.

6.746.3.17 `virtual bool decaf::net::Socket::getReuseAddress () const throw (SocketException)` [virtual]

Gets the reuse address flag, SO_REUSEADDR.

Returns

True if the address can be reused.

Exceptions

SocketException (p. 3298) if the operation fails.

6.746.3.18 `virtual int decaf::net::Socket::getSendBufferSize () const throw (SocketException)` [virtual]

Gets the send buffer size for this socket, SO_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

Returns

the size in bytes of the send buffer.

Exceptions

SocketException (p. 3298) if the operation fails.

6.746.3.19 `virtual int decaf::net::Socket::getSoLinger () const throw (SocketException)` [virtual]

Gets the linger time for the socket, SO_LINGER.

A return value of -1 indicates that the option is disabled.

Returns

The linger time in seconds.

Exceptions

SocketException (p. 3298) if the operation fails.

6.746.3.20 `virtual int decaf::net::Socket::getSoTimeout () const throw (SocketException)` [virtual]

Gets the timeout for socket operations, SO_TIMEOUT.

Returns

The timeout in milliseconds for socket operations.

Exceptions

SocketException (p. 3298) Thrown if unable to retrieve the information.

6.746.3.21 `virtual bool decaf::net::Socket::getTcpNoDelay () const throw (SocketException)` [virtual]

Gets the Status of the TCP_NODELAY setting for this socket.

Returns

true if TCP_NODELAY is enabled for the socket.

Exceptions

SocketException (p. 3298) Thrown if unable to set the information.

6.746.3.22 `virtual int decaf::net::Socket::getTrafficClass () const throw (SocketException)` [virtual]

Gets the Traffic Class setting for this **Socket** (p. 3281), sometimes referred to as Type of Service setting.

This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 3281) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

Returns

the bitset result of querying the traffic class setting.

Exceptions

SocketException (p. 3298) if an error is encountered while performing this operation.

6.746.3.23 `void decaf::net::Socket::initSocketImpl (const std::string & address, int port, const InetAddress * localAddress, int localPort) throw (decaf::io::IOException, decaf::net::UnknownHostException)` [protected]

6.746.3.24 `bool decaf::net::Socket::isBound () const` [inline]

Returns

true if this **Socket** (p. 3281) has been bound to a Local address.

6.746.3.25 `bool decaf::net::Socket::isClosed () const` [inline]

Returns

true if the **Socket** (p. 3281) has been closed.

6.746.3.26 `bool decaf::net::Socket::isConnected () const [inline]`

Indicates whether or not this socket is connected to an end point.

Returns

true if connected, false otherwise.

6.746.3.27 `bool decaf::net::Socket::isInputShutdown () const [inline]`

Returns

true if input on this **Socket** (p. 3281) has been shutdown.

6.746.3.28 `bool decaf::net::Socket::isOutputShutdown () const [inline]`

Returns

true if output on this **Socket** (p. 3281) has been shutdown.

6.746.3.29 `virtual void decaf::net::Socket::sendUrgentData (int data) throw (decaf::io::IOException) [virtual]`

Sends one byte of urgent data to the **Socket** (p. 3281).

Parameters

data The value to write as urgent data, only the lower eight bits are sent.

Exceptions

IOException if an I/O error occurs while performing this operation.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2682).

6.746.3.30 `virtual void decaf::net::Socket::setKeepAlive (bool keepAlive) throw (SocketException) [virtual]`

Enables/disables the keep alive flag for this socket, SO_KEEPALIVE.

Parameters

keepAlive If true, enables the flag.

Exceptions

SocketException (p. 3298) if the operation fails.

6.746.3.31 `virtual void decaf::net::Socket::setOOBInline (bool value) throw (SocketException) [virtual]`

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the Socket's InputStream, no notification is give.

Returns

true if OOBINLINE is enabled, false otherwise.

Exceptions

SocketException (p. 3298) if an error is encountered while performing this operation.

Reimplemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2684).

6.746.3.32 `virtual void decaf::net::Socket::setReceiveBufferSize (int size) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException) [virtual]`

Sets the receive buffer size for this socket, SO_RCVBUF.

Parameters

size Number of bytes to set the receive buffer to.

Exceptions

SocketException (p. 3298) if the operation fails.

IllegalArgumentException if the value is zero or negative.

6.746.3.33 `virtual void decaf::net::Socket::setReuseAddress (bool reuse) throw (SocketException) [virtual]`

Sets the reuse address flag, SO_REUSEADDR.

Parameters

reuse If true, sets the flag.

Exceptions

SocketException (p. 3298) if the operation fails.

6.746.3.34 `virtual void decaf::net::Socket::setSendBufferSize (int size) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException) [virtual]`

Gets the send buffer size for this socket, SO_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

Parameters

size The number of bytes to set the send buffer to, must be larger than zero.

Exceptions

SocketException (p. 3298) if the operation fails.

IllegalArgumentException if the value is zero or negative.

6.746.3.35 `static void decaf::net::Socket::setSocketImplFactory (SocketImplFactory * factory) throw (decaf::io::IOException, decaf::net::SocketException) [static]`

Sets the instance of a **SocketImplFactory** (p. 3314) that the **Socket** (p. 3281) class should use when new instances of this class are created.

This method is only allowed to be used once during the lifetime of the application.

Parameters

factory The instance of a **SocketImplFactory** (p. 3314) to use when new **Socket** (p. 3281) objects are created.

Exceptions

IOException if an I/O error occurs while performing this operation.

SocketException (p. 3298) if this method has already been called with a valid factory.

6.746.3.36 `virtual void decaf::net::Socket::setSoLinger (bool state, int timeout) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException) [virtual]`

Sets the linger time (SO_LINGER) using a specified time value, this limits of this value are platform specific.

Parameters

state The state of SO_LINGER, true is on.

timeout The linger time in seconds, must be non-negative.

Exceptions

SocketException (p. 3298) if the operation fails.

IllegalArgumentException if state is true and timeout is negative.

6.746.3.37 `virtual void decaf::net::Socket::setSoTimeout (int timeout) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException) [virtual]`

Sets the timeout for socket operations, SO_TIMEOUT.

A value of zero indicates that timeout is infinite for operations on this socket.

Parameters

timeout The timeout in milliseconds for socket operations.

Exceptions

SocketException (p. 3298) Thrown if unable to set the information.

IllegalArgumentException if the timeout value is negative.

6.746.3.38 `virtual void decaf::net::Socket::setTcpNoDelay (bool value) throw (SocketException) [virtual]`

Sets the Status of the TCP_NODELAY param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p. 3281).

Parameters

value The setting for the socket's TCP_NODELAY option, true to enable.

Exceptions

SocketException (p. 3298) Thrown if unable to set the information.

6.746.3.39 `virtual void decaf::net::Socket::setTrafficClass (int value) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException) [virtual]`

Gets the Traffic Class setting for this **Socket** (p. 3281), sometimes referred to as Type of Service setting.

This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 3281) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

Parameters

value The integer value representing the traffic class setting bitset.

Exceptions

SocketException (p. 3298) if an error is encountered while performing this operation.

IllegalArgumentException if the value is not in the range [0..255].

6.746.3.40 `virtual void decaf::net::Socket::shutdownInput () throw (decaf::io::IOException) [virtual]`

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

Exceptions

IOException if an I/O error occurs while performing this operation.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2685).

6.746.3.41 `virtual void decaf::net::Socket::shutdownOutput () throw (decaf::io::IOException) [virtual]`

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to `OuputStream::write` will throw an `IOException`.

Exceptions

IOException if an I/O error occurs while performing this operation.

Reimplemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2685).

6.746.3.42 `virtual std::string decaf::net::Socket::toString () const [virtual]`

Returns

a string representing this **Socket** (p. 3281).

6.746.4 Friends And Related Function Documentation

6.746.4.1 `friend class ServerSocket [friend]`

6.746.5 Field Documentation

6.746.5.1 `SocketImpl* decaf::net::Socket::impl [mutable, protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Socket.h`

6.747 decaf::net::SocketAddress Class Reference

Base class for protocol specific **Socket** (p. 3281) addresses.

```
#include <src/main/decaf/net/SocketAddress.h>
```

Inheritance diagram for `decaf::net::SocketAddress`:

Public Member Functions

- `virtual ~SocketAddress ()`

6.747.1 Detailed Description

Base class for protocol specific **Socket** (p. 3281) addresses. These classes provide an immutable address object that is used by the **Socket** (p. 3281) classes.

Since

1.0

6.747.2 Constructor & Destructor Documentation

6.747.2.1 virtual decaf::net::SocketAddress::~SocketAddress () [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketAddress.h`

6.748 decaf::net::SocketError Class Reference

Static utility class to simplify handling of error codes for socket operations.

```
#include <src/main/decaf/net/SocketError.h>
```

Static Public Member Functions

- static int **getErrorCode** ()
Gets the last error appropriate for the platform.
- static std::string **getErrorString** ()
Gets the string description for the last error.

6.748.1 Detailed Description

Static utility class to simplify handling of error codes for socket operations.

6.748.2 Member Function Documentation

6.748.2.1 static int decaf::net::SocketError::getErrorCode () [static]

Gets the last error appropriate for the platform.

6.748.2.2 static std::string decaf::net::SocketError::getErrorString () [static]

Gets the string description for the last error.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketError.h`

6.749 decaf::net::SocketException Class Reference

Exception for errors when manipulating sockets.

```
#include <src/main/decaf/net/SocketException.h>
```

Inheritance diagram for decaf::net::SocketException:

Public Member Functions

- **SocketException** () throw ()
- **SocketException** (const **lang::Exception** &ex) throw ()
- **SocketException** (const **SocketException** &ex) throw ()
- **SocketException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SocketException** (const std::exception *cause) throw ()
Constructor.
- **SocketException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **SocketException** * **clone** () const
Clones this exception.
- virtual ~**SocketException** () throw ()

6.749.1 Detailed Description

Exception for errors when manipulating sockets.

6.749.2 Constructor & Destructor Documentation

- 6.749.2.1** decaf::net::SocketException::SocketException () throw () [inline]
- 6.749.2.2** decaf::net::SocketException::SocketException (const lang::Exception &ex) throw () [inline]
- 6.749.2.3** decaf::net::SocketException::SocketException (const SocketException &ex) throw () [inline]
- 6.749.2.4** decaf::net::SocketException::SocketException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.749.2.5 decaf::net::SocketException::SocketException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.749.2.6 decaf::net::SocketException::SocketException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.749.2.7 virtual decaf::net::SocketException::~~SocketException () throw () [inline, virtual]

6.749.3 Member Function Documentation

6.749.3.1 virtual SocketException* decaf::net::SocketException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2005).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2689), **decaf::net::BindException** (p. 770), **decaf::net::ConnectException** (p. 1167), **decaf::net::NoRouteToHostException** (p. 2642), and **decaf::net::PortUnreachableException** (p. 2783).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketException.h**

6.750 decaf::net::SocketFactory Class Reference

The **SocketFactory** (p. 3301) is used to create **Socket** (p. 3281) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

```
#include <src/main/decaf/net/SocketFactory.h>
```

Inheritance diagram for decaf::net::SocketFactory:

Public Member Functions

- virtual **~SocketFactory** ()
- virtual **Socket * createSocket** () throw (decaf::io::IOException)
*Creates an unconnected **Socket** (p. 3281) object.*
- virtual **Socket * createSocket** (const **InetAddress** *host, int port)=0 throw (decaf::io::IOException, decaf::net::UnknownHostException)
*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*
- virtual **Socket * createSocket** (const **InetAddress** *host, int port, const **InetAddress** *ifAddress, int localPort)=0 throw (decaf::io::IOException, decaf::net::UnknownHostException)
*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*
- virtual **Socket * createSocket** (const std::string &name, int port)=0 throw (decaf::io::IOException, decaf::net::UnknownHostException)
*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*
- virtual **Socket * createSocket** (const std::string &name, int port, const **InetAddress** *ifAddress, int localPort)=0 throw (decaf::io::IOException, decaf::net::UnknownHostException)
*Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).*

Static Public Member Functions

- static **SocketFactory * getDefault** ()
*Returns an pointer to the default **SocketFactory** (p. 3301) for this Application, there is only one default **SocketFactory** (p. 3301) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 3301) class and in not to be deleted by the caller.*

Protected Member Functions

- **SocketFactory** ()

6.750.1 Detailed Description

The **SocketFactory** (p. 3301) is used to create **Socket** (p. 3281) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

See also

decaf.net.Socket (p. 3281)

Since

1.0

6.750.2 Constructor & Destructor Documentation

6.750.2.1 **decaf::net::SocketFactory::SocketFactory** () [protected]

6.750.2.2 **virtual decaf::net::SocketFactory::~~SocketFactory** () [virtual]

6.750.3 Member Function Documentation

6.750.3.1 **virtual Socket* decaf::net::SocketFactory::createSocket** () throw (**decaf::io::IOException**) [virtual]

Creates an unconnected **Socket** (p. 3281) object.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if the **Socket** (p. 3281) cannot be created.

Reimplemented in **decaf::internal::net::DefaultSocketFactory** (p. 1579), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1589), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2692).

6.750.3.2 **virtual Socket* decaf::net::SocketFactory::createSocket** (**const InetAddress * host**, **int port**) throw (**decaf::io::IOException**, **decaf::net::UnknownHostException**) [pure virtual]

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host to connect the socket to.

port The port on the remote host to connect to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1581), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1590), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2693).

6.750.3.3 `virtual Socket* decaf::net::SocketFactory::createSocket (const std::string & name, int port, const InetAddress * ifAddress, int localPort) throw (decaf::io::IOException, decaf::net::UnknownHostException) [pure virtual]`

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host name or IP address to connect the socket to.

port The port on the remote host to connect to.

ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.

localPort The local port to bind the **Socket** (p. 3281) to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1579), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1591), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2694).

6.750.3.4 `virtual Socket* decaf::net::SocketFactory::createSocket (const std::string & name, int port) throw (decaf::io::IOException, decaf::net::UnknownHostException) [pure virtual]`

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

Parameters

host The host name or IP address to connect the socket to.

port The port on the remote host to connect to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1580), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1591), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2693).

6.750.3.5 `virtual Socket* decaf::net::SocketFactory::createSocket (const InetAddress * host, int port, const InetAddress * ifAddress, int localPort) throw (decaf::io::IOException, decaf::net::UnknownHostException) [pure virtual]`

Creates a new **Socket** (p. 3281) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3301).

The **Socket** (p. 3281) will be bound to the specified local address and port.

Parameters

host The host to connect the socket to.

port The port on the remote host to connect to.

ifAddress The address on the local machine to bind the **Socket** (p. 3281) to.

localPort The local port to bind the **Socket** (p. 3281) to.

Returns

a new **Socket** (p. 3281) object, caller must free this object when done.

Exceptions

IOException if an I/O error occurs while creating the **Socket** (p. 3281) object.

UnknownHostException (p. 3649) if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1580), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1592), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2694).

6.750.3.6 `static SocketFactory* decaf::net::SocketFactory::getDefault () [static]`

Returns an pointer to the default **SocketFactory** (p. 3301) for this Application, there is only one default **SocketFactory** (p. 3301) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 3301) class and in not to be deleted by the caller.

Returns

pointer to the applications default **SocketFactory** (p. 3301).

Exceptions

SocketException (p. 3298) if an error occurs while getting the default instance.

Reimplemented in **decaf::net::ssl::SSLSocketFactory** (p. 3346).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketFactory.h`

6.751 decaf::internal::net::SocketFileDescriptor Class Reference

File Descriptor type used internally by Decaf Socket objects.

```
#include <src/main/decaf/internal/net/SocketFileDescriptor.h>
```

Inheritance diagram for `decaf::internal::net::SocketFileDescriptor`:

Public Member Functions

- **SocketFileDescriptor** (long value)
- virtual **~SocketFileDescriptor** ()
- long **getValue** () const
Gets the OS Level FileDescriptor.

6.751.1 Detailed Description

File Descriptor type used internally by Decaf Socket objects.

Since

1.0

6.751.2 Constructor & Destructor Documentation

6.751.2.1 `decaf::internal::net::SocketFileDescriptor::SocketFileDescriptor (long value)`

6.751.2.2 `virtual decaf::internal::net::SocketFileDescriptor::~~SocketFileDescriptor () [virtual]`

6.751.3 Member Function Documentation

6.751.3.1 `long decaf::internal::net::SocketFileDescriptor::getValue () const`

Gets the OS Level FileDescriptor.

Returns

a FileDescriptor value.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/SocketFileDescriptor.h`

6.752 decaf::net::SocketImpl Class Reference

Acts as a base class for all physical **Socket** (p. 3281) implementations.

```
#include <src/main/decaf/net/SocketImpl.h>
```

Inheritance diagram for decaf::net::SocketImpl:

Public Member Functions

- **SocketImpl** ()
- virtual **~SocketImpl** ()
- virtual void **create** ()=0 throw (decaf::io::IOException)
*Creates the underlying platform **Socket** (p. 3281) data structures which allows for **Socket** (p. 3281) options to be applied.*
- virtual void **accept** (**SocketImpl** *socket)=0 throw (decaf::io::IOException, decaf::net::SocketException, decaf::net::SocketTimeoutException)
*Accepts a new connection on the given **Socket** (p. 3281).*
- virtual void **connect** (const std::string &hostname, int **port**, int timeout)=0 throw (decaf::io::IOException, decaf::net::SocketTimeoutException, decaf::lang::exceptions::IllegalArgumentException)
Connects this socket to the given host and port.
- virtual void **bind** (const std::string &ipaddress, int **port**)=0 throw (decaf::io::IOException)
*Binds this **Socket** (p. 3281) instance to the local ip address and port number given.*
- virtual void **listen** (int backlog)=0 throw (decaf::io::IOException)
Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.
- virtual **decaf::io::InputStream** * **getInputStream** ()=0 throw (decaf::io::IOException)
*Gets the **InputStream** linked to this **Socket** (p. 3281).*
- virtual **decaf::io::OutputStream** * **getOutputStream** ()=0 throw (decaf::io::IOException)
*Gets the **OutputStream** linked to this **Socket** (p. 3281).*
- virtual int **available** ()=0 throw (decaf::io::IOException)
*Gets the number of bytes that can be read from the **Socket** (p. 3281) without blocking.*
- virtual void **close** ()=0 throw (decaf::io::IOException)
Closes the socket, terminating any blocked reads or writes.

- virtual void **shutdownInput** ()=0 throw (decaf::io::IOException)
Places the input stream for this socket at "end of stream".
- virtual void **shutdownOutput** ()=0 throw (decaf::io::IOException)
Disables the output stream for this socket.
- virtual int **getOption** (int option) const =0 throw (decaf::io::IOException)
*Gets the specified **Socket** (p. 3281) option.*
- virtual void **setOption** (int option, int value)=0 throw (decaf::io::IOException)
*Sets the specified option on the **Socket** (p. 3281) if supported.*
- int **getPort** () const
Gets the port that this socket has been assigned.
- int **getLocalPort** () const
Gets the value of this SocketImpl's local port field.
- std::string **getInetAddress** () const
Gets the value of this SocketImpl's address field.
- const decaf::io::FileDescriptor * **getFileDescriptor** () const
*Gets the FileDescriptor for this **Socket** (p. 3281), the Object is owned by this **Socket** (p. 3281) and should not be deleted by the caller.*
- virtual std::string **getLocalAddress** () const =0
*Gets the value of the local Inet address the **Socket** (p. 3281) is bound to if bound, otherwise return the **InetAddress** (p. 1884) ANY value "0.0.0.0".*
- std::string **toString** () const
*Returns a string containing the address and port of this **Socket** (p. 3281) instance.*
- virtual bool **supportsUrgentData** () const
- virtual void **sendUrgentData** (int data) throw (decaf::io::IOException)
*Sends on byte of urgent data to the **Socket** (p. 3281).*

Protected Attributes

- int **port**
*The remote port that this **Socket** (p. 3281) is connected to.*
- int **localPort**
*The port on the Local Machine that this **Socket** (p. 3281) is Bound to.*
- std::string **address**
*The Remote Address that the **Socket** (p. 3281) is connected to.*
- io::FileDescriptor * **fd**
*The File Descriptor for this **Socket** (p. 3281).*

6.752.1 Detailed Description

Acts as a base class for all physical **Socket** (p. 3281) implementations.

Since

1.0

6.752.2 Constructor & Destructor Documentation

6.752.2.1 decaf::net::SocketImpl::SocketImpl ()

6.752.2.2 virtual decaf::net::SocketImpl::~~SocketImpl () [virtual]

6.752.3 Member Function Documentation

6.752.3.1 virtual void decaf::net::SocketImpl::accept (SocketImpl * *socket*) throw (decaf::io::IOException, decaf::net::SocketException, decaf::net::SocketTimeoutException) [pure virtual]

Accepts a new connection on the given **Socket** (p. 3281).

Parameters

socket The accepted connection.

Exceptions

IOException if an I/O error occurs while attempting this operation.

SocketException (p. 3298) if an error occurs while performing an Accept on the socket.

SocketTimeoutException (p. 3319) if the accept call times out due to SO_TIMEOUT being set.

6.752.3.2 virtual int decaf::net::SocketImpl::available () throw (decaf::io::IOException) [pure virtual]

Gets the number of bytes that can be read from the **Socket** (p. 3281) without blocking.

Returns

the number of bytes that can be read from the **Socket** (p. 3281) without blocking.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3501).

6.752.3.3 virtual void decaf::net::SocketImpl::bind (const std::string & *ipaddress*, int *port*) throw (decaf::io::IOException) [pure virtual]

Binds this **Socket** (p. 3281) instance to the local ip address and port number given.

Parameters

ipaddress The address of local ip to bind to.
port The port number on the host to bind to.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implemented in `decaf::internal::net::tcp::TcpSocket` (p. 3501).

6.752.3.4 `virtual void decaf::net::SocketImpl::close () throw (decaf::io::IOException) [pure virtual]`

Closes the socket, terminating any blocked reads or writes.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implemented in `decaf::internal::net::tcp::TcpSocket` (p. 3501).

6.752.3.5 `virtual void decaf::net::SocketImpl::connect (const std::string & hostname, int port, int timeout) throw (decaf::io::IOException, decaf::net::SocketTimeoutException, decaf::lang::exceptions::IllegalArgumentException) [pure virtual]`

Connects this socket to the given host and port.

Parameters

hostname The name of the host to connect to, or IP address.
port The port number on the host to connect to.
timeout Time in milliseconds to wait for a connection, 0 indicates forever.

Exceptions

IOException if an I/O error occurs while attempting this operation.
SocketTimeoutException (p. 3319) if the connect call times out due to timeout being set.
IllegalArgumentException if a parameter has an illegal value.

Implemented in `decaf::internal::net::tcp::TcpSocket` (p. 3502).

6.752.3.6 `virtual void decaf::net::SocketImpl::create () throw (decaf::io::IOException) [pure virtual]`

Creates the underlying platform `Socket` (p. 3281) data structures which allows for `Socket` (p. 3281) options to be applied.

The created socket is in an unconnected state.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implemented in `decaf::internal::net::tcp::TcpSocket` (p. 3502).

6.752.3.7 `const decaf::io::FileDescriptor* decaf::net::SocketImpl::getFileDescriptor () const [inline]`

Gets the FileDescriptor for this **Socket** (p. 3281), the Object is owned by this **Socket** (p. 3281) and should not be deleted by the caller.

Returns

a pointer to this Socket's FileDescriptor object.

6.752.3.8 `std::string decaf::net::SocketImpl::getInetAddress () const [inline]`

Gets the value of this SocketImpl's address field.

Returns

the value of the address field.

6.752.3.9 `virtual decaf::io::InputStream* decaf::net::SocketImpl::getInputStream () throw (decaf::io::IOException) [pure virtual]`

Gets the InputStream linked to this **Socket** (p. 3281).

Returns

an InputStream pointer owned by the **Socket** (p. 3281) object.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3502).

6.752.3.10 `virtual std::string decaf::net::SocketImpl::getLocalAddress () const [pure virtual]`

Gets the value of the local Inet address the **Socket** (p. 3281) is bound to if bound, otherwise return the **InetAddress** (p. 1884) ANY value "0.0.0.0".

Returns

the local address bound to, or ANY.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3503).

6.752.3.11 `int decaf::net::SocketImpl::getLocalPort () const [inline]`

Gets the value of this SocketImpl's local port field.

Returns

the value of localPort.

6.752.3.12 `virtual int decaf::net::SocketImpl::getOption (int option) const
throw (decaf::io::IOException) [pure virtual]`

Gets the specified **Socket** (p. 3281) option.

Parameters

option The **Socket** (p. 3281) options whose value is to be retrieved.

Returns

the value of the given socket option.

Exceptions

IOException if an I/O error occurs while performing this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3503).

6.752.3.13 `virtual decaf::io::OutputStream* decaf::net::SocketImpl::getOutputStream () throw (decaf::io::IOException) [pure virtual]`

Gets the OutputStream linked to this **Socket** (p. 3281).

Returns

an OutputStream pointer owned by the **Socket** (p. 3281) object.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3503).

6.752.3.14 `int decaf::net::SocketImpl::getPort () const [inline]`

Gets the port that this socket has been assigned.

Returns

the Socket's port number.

6.752.3.15 `virtual void decaf::net::SocketImpl::listen (int backlog) throw (decaf::io::IOException) [pure virtual]`

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

Parameters

backlog The maximum length of the connection queue.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3504).

6.752.3.16 `virtual void decaf::net::SocketImpl::sendUrgentData (int data)
throw (decaf::io::IOException) [virtual]`

Sends on byte of urgent data to the **Socket** (p. 3281).

Parameters

data The value to write as urgent data, only the lower eight bits are sent.

Exceptions

IOException if an I/O error occurs while performing this operation.

6.752.3.17 `virtual void decaf::net::SocketImpl::setOption (int option, int value
) throw (decaf::io::IOException) [pure virtual]`

Sets the specified option on the **Socket** (p. 3281) if supported.

Parameters

option The **Socket** (p. 3281) option to set.

value The value of the socket option to apply to the socket.

Exceptions

IOException if an I/O error occurs while performing this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3505).

6.752.3.18 `virtual void decaf::net::SocketImpl::shutdownInput () throw (
decaf::io::IOException) [pure virtual]`

Places the input stream for this socket at "end of stream".

Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 3312) on the socket, the stream will return EOF.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3505).

6.752.3.19 `virtual void decaf::net::SocketImpl::shutdownOutput () throw (decaf::io::IOException)` [pure virtual]

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 3313) on the socket, the stream will throw an **IOException**.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3505).

6.752.3.20 `virtual bool decaf::net::SocketImpl::supportsUrgentData () const` [inline, virtual]

Returns

true if this **SocketImpl** (p. 3306) supports sending Urgent Data. The default implementation always returns false.

6.752.3.21 `std::string decaf::net::SocketImpl::toString () const`

Returns a string containing the address and port of this **Socket** (p. 3281) instance.

Returns

a string containing the address and port of this socket.

6.752.4 Field Documentation

6.752.4.1 `std::string decaf::net::SocketImpl::address` [protected]

The Remote Address that the **Socket** (p. 3281) is connected to.

6.752.4.2 `io::FileDescriptor* decaf::net::SocketImpl::fd` [protected]

The File Descriptor for this **Socket** (p. 3281).

6.752.4.3 `int decaf::net::SocketImpl::localPort` [protected]

The port on the Local Machine that this **Socket** (p. 3281) is Bound to.

6.752.4.4 `int decaf::net::SocketImpl::port` [protected]

The remote port that this **Socket** (p. 3281) is connected to.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketImpl.h`

6.753 decaf::net::SocketImplFactory Class Reference

Factory class interface for a Factory that creates SocketImpl objects.

```
#include <src/main/decaf/net/SocketImplFactory.h>
```

Public Member Functions

- virtual **~SocketImplFactory** ()
- virtual **SocketImpl * createSocketImpl** ()=0

*Creates a new SocketImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 3306).*

6.753.1 Detailed Description

Factory class interface for a Factory that creates SocketImpl objects. These factories can be used to create various types of Sockets, e.g. Streaming, Multicast, SSL, or platform specific variations of these types.

See also

decaf::net::Socket (p. 3281)
decaf::net::ServerSocket (p. 3136)

Since

1.0

6.753.2 Constructor & Destructor Documentation

6.753.2.1 virtual **decaf::net::SocketImplFactory::~~SocketImplFactory** ()
[inline, virtual]

6.753.3 Member Function Documentation

6.753.3.1 virtual **SocketImpl*** **decaf::net::SocketImplFactory::createSocketImpl** ()
[pure virtual]

Creates a new SocketImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 3306).

Returns

new **SocketImpl** (p. 3306) instance that is owned by the caller.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketImplFactory.h**

6.754 decaf::net::SocketOptions Class Reference

```
#include <src/main/decaf/net/SocketOptions.h>
```

Inheritance diagram for decaf::net::SocketOptions:

Public Member Functions

- virtual `~SocketOptions()`

Static Public Attributes

- static const int **SOCKET_OPTION_TCP_NODELAY**
Disable Nagle's algorithm for this connection.
- static const int **SOCKET_OPTION_BINDADDR**
Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).
- static const int **SOCKET_OPTION_REUSEADDR**
Sets SO_REUSEADDR for a socket.
- static const int **SOCKET_OPTION_BROADCAST**
Sets SO_BROADCAST for a socket.
- static const int **SOCKET_OPTION_IP_MULTICAST_IF**
Set which outgoing interface on which to send multicast packets.
- static const int **SOCKET_OPTION_IP_MULTICAST_IF2**
Same as above.
- static const int **SOCKET_OPTION_IP_MULTICAST_LOOP**
This option enables or disables local loopback of multicast datagrams.
- static const int **SOCKET_OPTION_IP_TOS**
This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.
- static const int **SOCKET_OPTION_LINGER**
Specify a linger-on-close timeout.
- static const int **SOCKET_OPTION_TIMEOUT**
*Set a timeout on blocking **Socket** (p. 3281) operations.*
- static const int **SOCKET_OPTION_SNDBUF**
Set a hint the size of the underlying buffers used by the platform for outgoing network I/O.
- static const int **SOCKET_OPTION_RCVBUF**

Set a hint the size of the underlying buffers used by the platform for incoming network I/O.

- static const int **SOCKET_OPTION_KEEPAKIVE**

When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer.

- static const int **SOCKET_OPTION_OOBNLINE**

When the OOBINLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream.

6.754.1 Detailed Description

Since

1.0

6.754.2 Constructor & Destructor Documentation

- 6.754.2.1** virtual decaf::net::SocketOptions::~SocketOptions () [virtual]

6.754.3 Field Documentation

- 6.754.3.1** const int decaf::net::SocketOptions::SOCKET_OPTION_BINDADDR [static]

Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).

The default local address of a socket is INADDR_ANY, meaning any local address on a multi-homed host. A multi-homed host can use this option to accept connections to only one of its addresses (in the case of a **ServerSocket** (p. 3136) or DatagramSocket), or to specify its return address to the peer (for a **Socket** (p. 3281) or DatagramSocket). The parameter of this option is an **InetAddress** (p. 1884).

- 6.754.3.2** const int decaf::net::SocketOptions::SOCKET_OPTION_BROADCAST [static]

Sets SO_BROADCAST for a socket.

This option enables and disables the ability of the process to send broadcast messages. It is supported for only datagram sockets and only on networks that support the concept of a broadcast message (e.g. Ethernet, token ring, etc.), and it is set by default for DatagramSockets.

- 6.754.3.3** const int decaf::net::SocketOptions::SOCKET_OPTION_IP_MULTICAST_IF [static]

Set which outgoing interface on which to send multicast packets.

Useful on hosts with multiple network interfaces, where applications want to use other than the system default. Takes/returns an **InetAddress** (p. 1884).

Valid for Multicast: DatagramSocketImpl.

6.754.3.4 `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_MULTICAST_IF2` [static]

Same as above.

This option is introduced so that the behaviour with `IP_MULTICAST_IF` will be kept the same as before, while this new option can support setting outgoing interfaces with either IPv4 and IPv6 addresses.

6.754.3.5 `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_MULTICAST_LOOP` [static]

This option enables or disables local loopback of multicast datagrams.

This option is enabled by default for Multicast Sockets.

6.754.3.6 `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_TOS` [static]

This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.

6.754.3.7 `const int decaf::net::SocketOptions::SOCKET_OPTION_KEEPALIVE` [static]

When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer.

This probe is a TCP segment to which the peer must respond. One of three responses is expected:

1. The peer responds with the expected ACK. The application is not notified (since everything is OK). TCP will send another probe following another 2 hours of inactivity.
2. The peer responds with an RST, which tells the local TCP that the peer host has crashed and rebooted. The socket is closed.
3. There is no response from the peer. The socket is closed.

The purpose of this option is to detect if the peer host crashes.

Valid only for TCP socket: **SocketImpl** (p. 3306)

6.754.3.8 `const int decaf::net::SocketOptions::SOCKET_OPTION_LINGER` [static]

Specify a linger-on-close timeout.

This option disables/enables immediate return from a `close()` of a TCP **Socket** (p. 3281). Enabling this option with a non-zero Integer timeout means that a `close()` will block pending the transmission and acknowledgment of all data written to the peer, at which point the socket is closed gracefully. Upon reaching the linger timeout, the socket is closed forcefully, with a TCP RST. Enabling the option with a timeout of zero does a forceful close immediately. If the specified timeout value exceeds 65,535 it will be reduced to 65,535.

Valid only for TCP: **SocketImpl** (p. 3306)

6.754.3.9 const int decaf::net::SocketOptions::SOCKET_OPTION_OOBLINE
[static]

When the OOBLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream.

When the option is disabled (which is the default) urgent data is silently discarded.

6.754.3.10 const int decaf::net::SocketOptions::SOCKET_OPTION_RCVBUF
[static]

Set a hint the size of the underlying buffers used by the platform for incoming network I/O.

When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be received over the socket. When used in get, this must return the size of the buffer actually used by the platform when receiving in data on this socket. Valid for all sockets: **SocketImpl** (p. 3306), **DatagramSocketImpl**.

6.754.3.11 const int decaf::net::SocketOptions::SOCKET_OPTION_REUSEADDR [static]

Sets SO_REUSEADDR for a socket.

This is used only for MulticastSockets in decaf, and it is set by default for MulticastSockets.

6.754.3.12 const int decaf::net::SocketOptions::SOCKET_OPTION_SNDBUF
[static]

Set a hint the size of the underlying buffers used by the platform for outgoing network I/O.

When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be sent over the socket. When used in get, this must return the size of the buffer actually used by the platform when sending out data on this socket. Valid for all sockets: **SocketImpl** (p. 3306), **DatagramSocketImpl**

6.754.3.13 const int decaf::net::SocketOptions::SOCKET_OPTION_TCP_NODELAY [static]

Disable Nagle's algorithm for this connection.

Written data to the network is not buffered pending acknowledgment of previously written data. Valid for TCP sockets.

6.754.3.14 const int decaf::net::SocketOptions::SOCKET_OPTION_TIMEOUT
[static]

Set a timeout on blocking **Socket** (p. 3281) operations.

The option must be set prior to entering a blocking operation to take effect.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketOptions.h**

6.755 decaf::net::SocketTimeoutException Class Reference

```
#include <src/main/decaf/net/SocketTimeoutException.h>
```

Inheritance diagram for decaf::net::SocketTimeoutException:

Public Member Functions

- **SocketTimeoutException** () throw ()
Default Constructor.
- **SocketTimeoutException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **SocketTimeoutException** (const **SocketTimeoutException** &ex) throw ()
Copy Constructor.
- **SocketTimeoutException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SocketTimeoutException** (const std::exception *cause) throw ()
Constructor.
- **SocketTimeoutException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **SocketTimeoutException** * **clone** () const
Clones this exception.
- virtual ~**SocketTimeoutException** () throw ()

6.755.1 Constructor & Destructor Documentation

6.755.1.1 decaf::net::SocketTimeoutException::SocketTimeoutException () throw () [inline]

Default Constructor.

6.755.1.2 decaf::net::SocketTimeoutException::SocketTimeoutException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.755.1.3 decaf::net::SocketTimeoutException::SocketTimeoutException (const SocketTimeoutException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.755.1.4 decaf::net::SocketTimeoutException::SocketTimeoutException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.755.1.5 decaf::net::SocketTimeoutException::SocketTimeoutException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.755.1.6 decaf::net::SocketTimeoutException::SocketTimeoutException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.755.1.7 `virtual decaf::net::SocketTimeoutException::~~SocketTimeoutException () throw () [inline, virtual]`

6.755.2 Member Function Documentation

6.755.2.1 `virtual SocketTimeoutException* decaf::net::SocketTimeoutException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::InterruptedIOException** (p.1992).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketTimeoutException.h`

6.756 decaf::net::ssl::SSLContext Class Reference

Represents an implementation of the Secure **Socket** (p.3281) Layer for streaming based sockets.

`#include <src/main/decaf/net/ssl/SSLContext.h>`

Public Member Functions

- **SSLContext** (**SSLContextSpi** *contextImpl)
- `virtual ~SSLContext ()`
- **SocketFactory** * **getSocketFactory ()**
*Returns an **SocketFactory** (p.3301) instance for use with this Context, the **SocketFactory** (p.3301) is owned by the Context and should not be deleted by the caller.*
- **ServerSocketFactory** * **getServerSocketFactory ()**
*Returns an **ServerSocketFactory** (p.3145) instance for use with this Context, the **ServerSocketFactory** (p.3145) is owned by the Context and should not be deleted by the caller.*
- **SSLParameters** * **getDefaultSSLParameters ()**
- **SSLParameters** * **getSupportedSSLParameters ()**

Static Public Member Functions

- `static SSLContext * getDefault ()`
*Gets the Default **SSLContext** (p.3321).*
- `static void setDefault (SSLContext *context)`
*Sets the default **SSLContext** (p.3321) to be returned from future calls to *getDefault*.*

6.756.1 Detailed Description

Represents on implementation of the Secure **Socket** (p. 3281) Layer for streaming based sockets. This class servers a a source of factories to be used to create new SSL **Socket** (p. 3281) instances.

Since

1.0

6.756.2 Constructor & Destructor Documentation

6.756.2.1 decaf::net::ssl::SSLContext::SSLContext (SSLContextSpi * *contextImpl*)

6.756.2.2 virtual decaf::net::ssl::SSLContext::~~SSLContext () [virtual]

6.756.3 Member Function Documentation

6.756.3.1 static SSLContext* decaf::net::ssl::SSLContext::getDefault () [static]

Gets the Default **SSLContext** (p. 3321).

The default instance of the **SSLContext** (p. 3321) should be immediately usable without any need for the client to initialize this context.

Returns

a pointer to the Default **SSLContext** (p. 3321) instance.

6.756.3.2 SSLParameters* decaf::net::ssl::SSLContext::getDefaultSSLParameters ()

Returns

a new instance of an **SSLParameters** (p. 3326) object containing the default set of settings for this **SSLContext** (p. 3321).

Exceptions

UnsupportedOperationException if the parameters cannot be retrieved.

6.756.3.3 ServerSocketFactory* decaf::net::ssl::SSLContext::getServerSocketFactory ()

Returns an **ServerSocketFactory** (p. 3145) instance for use with this Context, the **ServerSocketFactory** (p. 3145) is owned by the Context and should not be deleted by the caller.

Returns

a pointer to this SSLContext's **ServerSocketFactory** (p. 3145) for creating **SSLServerSocket** (p. 3329) objects.

Exceptions

IllegalStateException if the **SSLContextSpi** (p. 3324) requires initialization but it has not yet been initialized.

6.756.3.4 **SocketFactory*** `decaf::net::ssl::SSLContext::getSocketFactory ()`

Returns an **SocketFactory** (p. 3301) instance for use with this Context, the **SocketFactory** (p. 3301) is owned by the Context and should not be deleted by the caller.

Returns

a pointer to this SSLContext's **SocketFactory** (p. 3301) for creating **SSLSocket** (p. 3337) objects.

Exceptions

IllegalStateException if the **SSLContextSpi** (p. 3324) requires initialization but it has not yet been initialized.

6.756.3.5 **SSLParameters*** `decaf::net::ssl::SSLContext::getSupportedSSLParameters ()`

Returns

a new instance of an **SSLParameters** (p. 3326) object containing the complete set of settings for this **SSLContext** (p. 3321).

Exceptions

UnsupportedOperationException if the parameters cannot be retrieved.

6.756.3.6 `static void decaf::net::ssl::SSLContext::setDefault (SSLContext * context) [static]`

Sets the default **SSLContext** (p. 3321) to be returned from future calls to `getDefault`.

The set **SSLContext** (p. 3321) must be fully initialized and usable. The caller is responsible for deleting this object before the Library shutdown methods are called.

Exceptions

NullPointerException if the context passed is NULL.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLContext.h`

6.757 decaf::net::ssl::SSLContextSpi Class Reference

Defines the interface that should be provided by an **SSLContext** (p. 3321) provider.

```
#include <src/main/decaf/net/ssl/SSLContextSpi.h>
```

Inheritance diagram for decaf::net::ssl::SSLContextSpi:

Public Member Functions

- virtual **~SSLContextSpi** ()
- virtual void **providerInit** (security::SecureRandom *random)=0
Perform the initialization of this Context.
- virtual **SSLParameters** * **providerGetDefaultSSLParameters** ()
*Creates a returns a new **SSLParameters** (p. 3326) instance that contains the default settings for this Providers **SSLContext** (p. 3321).*
- virtual **SSLParameters** * **providerGetSupportedSSLParameters** ()
*Creates and returns a new **SSLParameters** (p. 3326) instance that contains the full set of supported parameters for this SSL Context.*
- virtual **SocketFactory** * **providerGetSocketFactory** ()=0
*Returns a **SocketFactory** (p. 3301) instance that can be used to create new **SSLSocket** (p. 3337) objects.*
- virtual **ServerSocketFactory** * **providerGetServerSocketFactory** ()=0
*Returns a **ServerSocketFactory** (p. 3145) instance that can be used to create new **SSLServerSocket** (p. 3329) objects.*

6.757.1 Detailed Description

Defines the interface that should be provided by an **SSLContext** (p. 3321) provider.

Since

1.0

6.757.2 Constructor & Destructor Documentation

6.757.2.1 virtual decaf::net::ssl::SSLContextSpi::~SSLContextSpi () [virtual]

6.757.3 Member Function Documentation

6.757.3.1 virtual **SSLParameters*** decaf::net::ssl::SSLContextSpi::providerGetDefaultSSLParameters () [virtual]

Creates a returns a new **SSLParameters** (p. 3326) instance that contains the default settings for this Providers **SSLContext** (p. 3321).

The returned **SSLParameters** (p. 3326) instance is requires to have non-empty values in its ciphersuites and protocols.

Returns

new **SSLParameters** (p. 3326) instance with the **SSLContext** (p. 3321) defaults.

Exceptions

UnsupportedOperationException if the defaults cannot be obtained.

6.757.3.2 `virtual ServerSocketFactory* decaf::net::ssl::SSLContextSpi::providerGetServerSocketFactory ()`
[pure virtual]

Returns a **ServerSocketFactory** (p. 3145) instance that can be used to create new **SSLServerSocket** (p. 3329) objects.

The **ServerSocketFactory** (p. 3145) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 3301) instance that can be used to create new SSLServerSockets.

Exceptions

IllegalStateException if the **SSLContextSpi** (p. 3324) object requires initialization but has not been initialized yet.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2660).

6.757.3.3 `virtual SocketFactory* decaf::net::ssl::SSLContextSpi::providerGetSocketFactory ()`
[pure virtual]

Returns a **SocketFactory** (p. 3301) instance that can be used to create new **SSLSocket** (p. 3337) objects.

The **SocketFactory** (p. 3301) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 3301) instance that can be used to create new SSLSockets.

Exceptions

IllegalStateException if the **SSLContextSpi** (p. 3324) object requires initialization but has not been initialized yet.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2660).

6.757.3.4 virtual `SSLParameters*` `decaf::net::ssl::SSLContextSpi::providerGetSupportedSSLParameters ()` [virtual]

Creates and returns a new **SSLParameters** (p. 3326) instance that contains the full set of supported parameters for this SSL Context.

The returned **SSLParameters** (p. 3326) instance is requires to have non-empty values in its ciphersuites and protocols.

Returns

a new **SSLParameters** (p. 3326) instance with the full set of settings that are supported.

Exceptions

UnsupportedOperationException if the supported parameters cannot be obtained.

6.757.3.5 virtual void `decaf::net::ssl::SSLContextSpi::providerInit (security::SecureRandom * random)` [pure virtual]

Perform the initialization of this Context.

Parameters

random Pointer to an instance of a secure random number generator.

Exceptions

NullPointerException if the SecureRandom instance is NULL.

KeyManagementException if an error occurs while initializing the context.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLContextSpi` (p. 2661).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLContextSpi.h`

6.758 decaf::net::ssl::SSLParameters Class Reference

```
#include <src/main/decaf/net/ssl/SSLParameters.h>
```

Public Member Functions

- **SSLParameters** ()
*Creates a new **SSLParameters** (p. 3326) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.*
- **SSLParameters** (const std::vector< std::string > &cipherSuites)
*Creates a new **SSLParameters** (p. 3326) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.*

- **SSLParameters** (const std::vector< std::string > &cipherSuites, const std::vector< std::string > &protocols)

*Creates a new **SSLParameters** (p. 3326) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.*

- virtual ~**SSLParameters** ()
- std::vector< std::string > **getCipherSuites** () const
- void **setCipherSuites** (const std::vector< std::string > &cipherSuites)
Sets the vector of ciphersuites.

- std::vector< std::string > **getProtocols** () const
- void **setProtocols** (const std::vector< std::string > &protocols)
Sets the vector of protocols.

- bool **getWantClientAuth** () const
- void **setWantClientAuth** (bool wantClientAuth)
Sets whether client authentication should be requested.

- bool **getNeedClientAuth** () const
- void **setNeedClientAuth** (bool needClientAuth)
Sets whether client authentication should be required.

6.758.1 Constructor & Destructor Documentation

6.758.1.1 decaf::net::ssl::SSLParameters::SSLParameters ()

Creates a new **SSLParameters** (p. 3326) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.

6.758.1.2 decaf::net::ssl::SSLParameters::SSLParameters (const std::vector< std::string > & cipherSuites)

Creates a new **SSLParameters** (p. 3326) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.

Parameters

cipherSuites The vector of cipherSuites for this **SSLParameters** (p. 3326) instance (can be empty).

6.758.1.3 decaf::net::ssl::SSLParameters::SSLParameters (const std::vector< std::string > & cipherSuites, const std::vector< std::string > & protocols)

Creates a new **SSLParameters** (p. 3326) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.

Parameters

cipherSuites The vector of cipherSuites for this **SSLParameters** (p. 3326) instance (can be empty).

protocols The vector of protocols for this **SSLParameters** (p. 3326) instance (can be empty).

6.758.1.4 virtual decaf::net::ssl::SSLParameters::~~SSLParameters () [virtual]

6.758.2 Member Function Documentation

6.758.2.1 std::vector<std::string> decaf::net::ssl::SSLParameters::getCipherSuites () const [inline]

Returns

a copy of the vector of ciphersuites or an empty vector if none have been set.

6.758.2.2 bool decaf::net::ssl::SSLParameters::getNeedClientAuth () const [inline]

Returns

whether client authentication should be required.

6.758.2.3 std::vector<std::string> decaf::net::ssl::SSLParameters::getProtocols () const [inline]

Returns

a copy of the vector of protocols or an empty vector if none have been set.

6.758.2.4 bool decaf::net::ssl::SSLParameters::getWantClientAuth () const [inline]

Returns

whether client authentication should be requested.

6.758.2.5 void decaf::net::ssl::SSLParameters::setCipherSuites (const std::vector<std::string> & *cipherSuites*) [inline]

Sets the vector of ciphersuites.

Parameters

cipherSuites The vector of cipherSuites (can be an empty vector).

6.758.2.6 `void decaf::net::ssl::SSLParameters::setNeedClientAuth (bool needClientAuth) [inline]`

Sets whether client authentication should be required.

Calling this method clears the wantClientAuth flag.

Parameters

needClientAuth whether client authentication should be required.

6.758.2.7 `void decaf::net::ssl::SSLParameters::setProtocols (const std::vector< std::string > & protocols) [inline]`

Sets the vector of protocols.

Parameters

protocols the vector of protocols (or an empty vector)

6.758.2.8 `void decaf::net::ssl::SSLParameters::setWantClientAuth (bool wantClientAuth) [inline]`

Sets whether client authentication should be requested.

Calling this method clears the needClientAuth flag.

Parameters

whether client authentication should be requested.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLParameters.h`

6.759 decaf::net::ssl::SSLServerSocket Class Reference

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

`#include <src/main/decaf/net/ssl/SSLServerSocket.h>`

Inheritance diagram for `decaf::net::ssl::SSLServerSocket`:

Public Member Functions

- `virtual ~SSLServerSocket ()`
- `virtual std::vector< std::string > getSupportedCipherSuites () const =0`

*Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3329).*

- virtual std::vector< std::string > **getSupportedProtocols** () const =0
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3329) instance.*
- virtual std::vector< std::string > **getEnabledCipherSuites** () const =0
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3329).*
- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)=0
*Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3329) connection.*
- virtual std::vector< std::string > **getEnabledProtocols** () const =0
*Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3329).*
- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)=0
*Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3329) connection.*
- virtual bool **getWantClientAuth** () const =0
- virtual void **setWantClientAuth** (bool value)=0
*Sets whether or not this **Socket** (p. 3281) will request Client Authentication.*
- virtual bool **getNeedClientAuth** () const =0
- virtual void **setNeedClientAuth** (bool value)=0
*Sets whether or not this **Socket** (p. 3281) will require Client Authentication.*

Protected Member Functions

- **SSLServerSocket** ()
Creates a non-bound server socket.
- **SSLServerSocket** (int port)
*Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **SSLServerSocket** (int port, int backlog)
*Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **SSLServerSocket** (int port, int backlog, const decaf::net::InetAddress *address)
*Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.*

6.759.1 Detailed Description

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol. The main function of this class is to create **SSLSocket** (p. 3337) objects by accepting connections from client sockets over SSL.

Since

1.0

6.759.2 Constructor & Destructor Documentation

6.759.2.1 `decaf::net::ssl::SSLServerSocket::SSLServerSocket ()` [protected]

Creates a non-bound server socket.

6.759.2.2 `decaf::net::ssl::SSLServerSocket::SSLServerSocket (int port)` [protected]

Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 3314) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3306) is created.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

Exceptions

IOException if there is an I/O error while performing this operation.

IllegalArgumentException if the port value is negative or greater than 65535.

6.759.2.3 `decaf::net::ssl::SSLServerSocket::SSLServerSocket (int port, int backlog)` [protected]

Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3314) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3306) is created.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The the number of incoming connection attempts to queue before connections are refused.

Exceptions

IOException if there is an I/O error while performing this operation.

IllegalArgumentException if the port value is negative or greater than 65535.

6.759.2.4 decaf::net::ssl::SSLServerSocket::SSLServerSocket (int *port*, int *backlog*, const decaf::net::InetAddress * *address*) [protected]

Creates a new **ServerSocket** (p. 3136) bound to the specified port, if the value of port is 0, then any free port is chosen.

If the value of the ifAddress is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3314) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 3306) is created.

Parameters

port The port to bind the **ServerSocket** (p. 3136) to.

backlog The the number of incoming connection attempts to queue before connections are refused.

ifAddress The IP Address to bind to on the local machine.

Exceptions

IOException if there is an I/O error while performing this operation.

IllegalArgumentException if the port value is negative or greater than 65535.

6.759.2.5 virtual decaf::net::ssl::SSLServerSocket::~~SSLServerSocket () [virtual]

6.759.3 Member Function Documentation

6.759.3.1 virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getEnabledCipherSuites () const [pure virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3329).

Returns

vector of the names of all enabled Cipher Suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2666).

6.759.3.2 `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getEnabledProtocols () const [pure virtual]`

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3329).

Returns

vector of the names of all enabled Protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2667).

6.759.3.3 `virtual bool decaf::net::ssl::SSLServerSocket::getNeedClientAuth () const [pure virtual]`

Returns

true if the **Socket** (p. 3281) requires client Authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2667).

6.759.3.4 `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getSupportedCipherSuites () const [pure virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3329).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3281).

Returns

a vector containing the names of all the supported cipher suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2667).

6.759.3.5 `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getSupportedProtocols () const [pure virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3329) instance.

Returns

a vector containing the names of all the supported protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2667).

6.759.3.6 virtual bool decaf::net::ssl::SSLServerSocket::getWantClientAuth ()
const [pure virtual]

Returns

true if the **Socket** (p. 3281) request client Authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2668).

6.759.3.7 virtual void decaf::net::ssl::SSLServerSocket::setEnabledCipherSuites (
const std::vector< std::string > & *suites*) [pure virtual]

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3329) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

suites An Vector of names for all the Cipher Suites that are to be enabled.

Exceptions

IllegalArgumentException if the vector is empty or one of the names is invalid.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2668).

6.759.3.8 virtual void decaf::net::ssl::SSLServerSocket::setEnabledProtocols (
const std::vector< std::string > & *protocols*) [pure virtual]

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3329) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

protocols An Vector of names for all the Protocols that are to be enabled.

Exceptions

IllegalArgumentException if the vector is empty or one of the names is invalid.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2668).

6.759.3.9 virtual void decaf::net::ssl::SSLServerSocket::setNeedClientAuth (bool
value) [pure virtual]

Sets whether or not this **Socket** (p. 3281) will require Client Authentication.

If set to true the **Socket** (p. 3281) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

Parameters

value Whether the server socket should require client authentication.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLServerSocket` (p. 2668).

6.759.3.10 `virtual void decaf::net::ssl::SSLServerSocket::setWantClientAuth (bool value)` [pure virtual]

Sets whether or not this **Socket** (p. 3281) will request Client Authentication.

If set to true the **Socket** (p. 3281) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

Parameters

value Whether the server socket should request client authentication.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLServerSocket` (p. 2669).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLServerSocket.h`

6.760 `decaf::net::ssl::SSLServerSocketFactory` Class Reference

Factory class interface that provides methods to create SSL Server Sockets.

```
#include <src/main/decaf/net/ssl/SSLServerSocketFactory.h>
```

Inheritance diagram for `decaf::net::ssl::SSLServerSocketFactory`:

Public Member Functions

- `virtual ~SSLServerSocketFactory ()`
- `virtual std::vector< std::string > getDefaultCipherSuites ()=0`
Returns the list of cipher suites which are enabled by default.
- `virtual std::vector< std::string > getSupportedCipherSuites ()=0`
Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Static Public Member Functions

- `static ServerSocketFactory * getDefault ()`
*Returns the current default SSL **ServerSocketFactory** (p. 3145), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

Protected Member Functions

- `SSLServerSocketFactory` ()

6.760.1 Detailed Description

Factory class interface that provides methods to create SSL Server Sockets.

Since

1.0

6.760.2 Constructor & Destructor Documentation

6.760.2.1 `decaf::net::ssl::SSLServerSocketFactory::SSLServerSocketFactory ()`
[protected]

6.760.2.2 `virtual decaf::net::ssl::SSLServerSocketFactory::~~SSLServerSocketFactory ()` [virtual]

6.760.3 Member Function Documentation

6.760.3.1 `static ServerSocketFactory* decaf::net::ssl::SSLServerSocketFactory::getDefault ()`
[static]

Returns the current default SSL `ServerSocketFactory` (p.3145), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.

This method returns `SSLContext::getDefault()` (p.3322)->`getServerSocketFactory()`. If that call fails, a non-functional factory is returned.

Returns

the default SSL `ServerSocketFactory` (p.3145) pointer.

See also

`decaf::net::ssl::SSLContext::getDefault()` (p.3322)

Reimplemented from `decaf::net::ServerSocketFactory` (p.3148).

6.760.3.2 `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocketFactory::getDefaultCipherSuites ()`
[pure virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

`getSupportedCipherSuites()` (p. 3336)

Implemented in `decaf::internal::net::ssl::DefaultSSLServerSocketFactory` (p. 1586), and `decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory` (p. 2673).

6.760.3.3 `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocketFactory::getSupportedCipherSuites ()`
[pure virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

`getDefaultCipherSuites()` (p. 3336)

Implemented in `decaf::internal::net::ssl::DefaultSSLServerSocketFactory` (p. 1586), and `decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory` (p. 2673).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLServerSocketFactory.h`

6.761 decaf::net::ssl::SSLSocket Class Reference

```
#include <src/main/decaf/net/ssl/SSLSocket.h>
```

Inheritance diagram for `decaf::net::ssl::SSLSocket`:

Public Member Functions

- **SSLSocket** ()
- **SSLSocket** (const **InetAddress** *address, int port)
*Creates a new **SSLSocket** (p. 3337) instance and connects it to the given address and port.*
- **SSLSocket** (const **InetAddress** *address, int port, const **InetAddress** *localAddress, int localPort)
*Creates a new **SSLSocket** (p. 3337) instance and connects it to the given address and port.*
- **SSLSocket** (const std::string &host, int port)
*Creates a new **SSLSocket** (p. 3337) instance and connects it to the given host and port.*

- **SSLSocket** (const std::string &host, int port, const **InetAddress** *localAddress, int localPort)
*Creates a new **SSLSocket** (p. 3337) instance and connects it to the given host and port.*
- virtual ~**SSLSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () const =0
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3337).*
- virtual std::vector< std::string > **getSupportedProtocols** () const =0
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3337) instance.*
- virtual std::vector< std::string > **getEnabledCipherSuites** () const =0
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3281).*
- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)=0
*Sets the Cipher Suites that are to be enabled on the **SSL Socket** (p. 3281) connection.*
- virtual std::vector< std::string > **getEnabledProtocols** () const =0
*Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3281).*
- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)=0
*Sets the Protocols that are to be enabled on the **SSL Socket** (p. 3281) connection.*
- virtual **SSLParameters** **getSSLParameters** () const
*Returns an **SSLParameters** (p. 3326) object for this **SSLSocket** (p. 3337) instance.*
- virtual void **setSSLParameters** (const **SSLParameters** &value)
*Sets the **SSLParameters** (p. 3326) for this **SSLSocket** (p. 3337) using the supplied **SSLParameters** (p. 3326) instance.*
- virtual void **startHandshake** ()=0
*Initiates a handshake for this **SSL Connection**, this can be necessary for several reasons such as using new encryption keys, or starting a new session.*
- virtual void **setUseClientMode** (bool value)=0
Determines the mode that the socket uses when a handshake is initiated, client or server.
- virtual bool **getUseClientMode** () const =0
*Gets whether this **Socket** (p. 3281) is in Client or Server mode, true indicates that the mode is set to Client.*
- virtual void **setNeedClientAuth** (bool value)=0
*Sets the **Socket** (p. 3281) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*
- virtual bool **getNeedClientAuth** () const =0

Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.

- virtual void **setWantClientAuth** (bool value)=0

*Sets the **Socket** (p. 3281) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

- virtual bool **getWantClientAuth** () const =0

Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

6.761.1 Detailed Description

Since

1.0

6.761.2 Constructor & Destructor Documentation

6.761.2.1 decaf::net::ssl::SSLSocket::SSLSocket ()

6.761.2.2 decaf::net::ssl::SSLSocket::SSLSocket (const InetAddress * *address*, int *port*)

Creates a new **SSLSocket** (p. 3337) instance and connects it to the given address and port.

If the host parameter is empty then the loop back address is used.

Parameters

address The address to connect to.

port The port number to connect to [0...65535]

Exceptions

UnknownHostException (p. 3649) if the host cannot be resolved.

IOException if an I/O error occurs while connecting the **Socket** (p. 3281).

NullPointerException if the **InetAddress** (p. 1884) instance is NULL.

IllegalArgumentException if the port is not in range [0...65535]

6.761.2.3 decaf::net::ssl::SSLSocket::SSLSocket (const InetAddress * *address*, int *port*, const InetAddress * *localAddress*, int *localPort*)

Creates a new **SSLSocket** (p. 3337) instance and connects it to the given address and port.

The **Socket** (p. 3281) will also bind to the local address and port specified.

Parameters

address The address to connect to.

port The port number to connect to [0...65535]

localAddress The IP address on the local machine to bind to.

localPort The port on the local machine to bind to.

Exceptions

UnknownHostException (p. 3649) if the host cannot be resolved.

IOException if an I/O error occurs while connecting the **Socket** (p. 3281).

NullPointerException if the **InetAddress** (p. 1884) instance is NULL.

IllegalArgumentException if the port is not in range [0...65535]

6.761.2.4 decaf::net::ssl::SSLSocket::SSLSocket (const std::string & host, int port)

Creates a new **SSLSocket** (p. 3337) instance and connects it to the given host and port.

If the host parameter is empty then the loop back address is used.

Parameters

host The host name or IP address to connect to, empty string means loopback.

port The port number to connect to [0...65535]

Exceptions

UnknownHostException (p. 3649) if the host cannot be resolved.

IOException if an I/O error occurs while connecting the **Socket** (p. 3281).

IllegalArgumentException if the port is not in range [0...65535]

6.761.2.5 decaf::net::ssl::SSLSocket::SSLSocket (const std::string & host, int port, const InetAddress * localAddress, int localPort)

Creates a new **SSLSocket** (p. 3337) instance and connects it to the given host and port.

If the host parameter is empty then the loop back address is used.

Parameters

host The host name or IP address to connect to, empty string means loopback.

port The port number to connect to [0...65535]

localAddress The IP address on the local machine to bind to.

localPort The port on the local machine to bind to.

Exceptions

UnknownHostException (p. 3649) if the host cannot be resolved.

IOException if an I/O error occurs while connecting the **Socket** (p. 3281).

IllegalArgumentException if the port is not in range [0...65535]

6.761.2.6 `virtual decaf::net::ssl::SSLSocket::~~SSLSocket () [virtual]`

6.761.3 Member Function Documentation

6.761.3.1 `virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getEnabledCipherSuites () const [pure virtual]`

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3281).

Returns

vector of the names of all enabled Cipher Suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2679).

6.761.3.2 `virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getEnabledProtocols () const [pure virtual]`

Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3281).

Returns

vector of the names of all enabled Protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2680).

6.761.3.3 `virtual bool decaf::net::ssl::SSLSocket::getNeedClientAuth () const [pure virtual]`

Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2680).

6.761.3.4 `virtual SSLParameters decaf::net::ssl::SSLSocket::getSSLParameters () const [virtual]`

Returns an **SSLParameters** (p. 3326) object for this **SSLSocket** (p. 3337) instance.

The cipherSuites and protocols vectors in the returned **SSLParameters** (p. 3326) reference will never be empty.

Returns

an **SSLParameters** (p. 3326) object with the settings in use for the **SSLSocket** (p. 3337).

6.761.3.5 `virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getSupportedCipherSuites () const [pure virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3337).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3281).

Returns

a vector containing the names of all the supported cipher suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2681).

6.761.3.6 `virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getSupportedProtocols () const [pure virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3337) instance.

Returns

a vector containing the names of all the supported protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2681).

6.761.3.7 `virtual bool decaf::net::ssl::SSLSocket::getUseClientMode () const [pure virtual]`

Gets whether this **Socket** (p. 3281) is in Client or Server mode, true indicates that the mode is set to Client.

Returns

true if the **Socket** (p. 3281) is in Client mode, false otherwise.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2681).

6.761.3.8 `virtual bool decaf::net::ssl::SSLSocket::getWantClientAuth () const [pure virtual]`

Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2682).

6.761.3.9 `virtual void decaf::net::ssl::SSLSocket::setEnabledCipherSuites (const std::vector< std::string > & suites) [pure virtual]`

Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 3281) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

suites An Vector of names for all the Cipher Suites that are to be enabled.

Exceptions

IllegalArgumentException if the vector is empty or one of the names is invalid.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2683).

6.761.3.10 `virtual void decaf::net::ssl::SSLSocket::setEnabledProtocols (const std::vector< std::string > & protocols) [pure virtual]`

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 3281) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

protocols An Vector of names for all the Protocols that are to be enabled.

Exceptions

IllegalArgumentException if the vector is empty or one of the names is invalid.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2683).

6.761.3.11 `virtual void decaf::net::ssl::SSLSocket::setNeedClientAuth (bool value) [pure virtual]`

Sets the **Socket** (p. 3281) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

Parameters

value The value indicating if a client is required to authenticate itself or not.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2683).

6.761.3.12 virtual void decaf::net::ssl::SSLSocket::setSSLParameters (const SSLParameters & value) [virtual]

Sets the **SSLParameters** (p. 3326) for this **SSLSocket** (p. 3337) using the supplied **SSLParameters** (p. 3326) instance.

If the cipherSuites vector in the **SSLParameters** (p. 3326) instance is not empty then the setEnabledCipherSuites method is called with that vector, if the protocols vector in the **SSLParameters** (p. 3326) instance is not empty then the setEnabledProtocols method is called with that vector. If the needClientAuth value or the wantClientAuth value is true then the setNeedClientAuth and setWantClientAuth methods are called respectively with a value of true, otherwise the setWantClientAuth method is called with a value of false.

Parameters

value The **SSLParameters** (p. 3326) instance that is used to update this **SSLSocket**'s settings.

Exceptions

IllegalArgumentException if an error occurs while calling setEnabledCipherSuites or setEnabledProtocols.

6.761.3.13 virtual void decaf::net::ssl::SSLSocket::setUseClientMode (bool value) [pure virtual]

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 3281), once a handshake has been initiated this socket remains in the set mode; client or server, for the life of this object.

Parameters

value The mode setting, true for client or false for server.

Exceptions

IllegalArgumentException if the handshake process has begun and mode is locked.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2684).

6.761.3.14 virtual void decaf::net::ssl::SSLSocket::setWantClientAuth (bool value) [pure virtual]

Sets the **Socket** (p. 3281) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled and the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid then the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the setNeedClientAuth method.

Parameters

value The value indicating if a client is requested to authenticate itself or not.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2684).

6.761.3.15 `virtual void decaf::net::ssl::SSLSocket::startHandshake ()` [pure virtual]

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not required to support multiple handshakes and can throw an `IOException` to indicate an error.

Exceptions

IOException if an I/O error occurs while performing the Handshake

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2685).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLSocket.h`

6.762 decaf::net::ssl::SSLSocketFactory Class Reference

Factory class interface for a `SocketFactory` (p. 3301) that can create `SSLSocket` (p. 3337) objects.

```
#include <src/main/decaf/net/ssl/SSLSocketFactory.h>
```

Inheritance diagram for `decaf::net::ssl::SSLSocketFactory`:

Public Member Functions

- `virtual ~SSLSocketFactory ()`
- `virtual std::vector< std::string > getDefaultCipherSuites ()=0`
Returns the list of cipher suites which are enabled by default.
- `virtual std::vector< std::string > getSupportedCipherSuites ()=0`
Returns the names of the cipher suites which could be enabled for use on an SSL connection.
- `virtual Socket * createSocket (Socket *socket, std::string host, int port, bool autoClose)=0`
Returns a socket layered over an existing socket connected to the named host, at the given port.

Static Public Member Functions

- static **SocketFactory** * **getDefault** ()

*Returns the current default SSL **SocketFactory** (p. 3301), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

Protected Member Functions

- **SSLSocketFactory** ()

6.762.1 Detailed Description

Factory class interface for a **SocketFactory** (p. 3301) that can create **SSLSocket** (p. 3337) objects.

Since

1.0

6.762.2 Constructor & Destructor Documentation

6.762.2.1 decaf::net::ssl::SSLSocketFactory::SSLSocketFactory () [protected]

6.762.2.2 virtual decaf::net::ssl::SSLSocketFactory::~~SSLSocketFactory ()
[virtual]

6.762.3 Member Function Documentation

6.762.3.1 virtual **Socket*** decaf::net::ssl::SSLSocketFactory::createSocket (**Socket** * *socket*, std::string *host*, int *port*, bool *autoClose*) [pure virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

socket The existing socket to layer over.

host The server host the original **Socket** (p. 3281) is connected to.

port The server port the original **Socket** (p. 3281) is connected to.

autoClose Should the layered over **Socket** (p. 3281) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 3281) instance that wraps the given **Socket** (p. 3281).

Exceptions

IOException if an I/O exception occurs while performing this operation.

UnknownHostException (p. 3649) if the host is unknown.

Implemented in `decaf::internal::net::ssl::DefaultSSLSocketFactory` (p.1590), and `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory` (p.2692).

6.762.3.2 `static SocketFactory* decaf::net::ssl::SSLSocketFactory::getDefault ()`
[static]

Returns the current default SSL `SocketFactory` (p.3301), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.

This method returns `SSLContext::getDefault()` (p.3322)->`getSocketFactory()`. If that call fails, a non-functional factory is returned.

Returns

the default SSL `SocketFactory` (p.3301) pointer.

See also

`decaf::net::ssl::SSLContext::getDefault()` (p.3322)

Reimplemented from `decaf::net::SocketFactory` (p.3304).

6.762.3.3 `virtual std::vector<std::string> decaf::net::ssl::SSLSocketFactory::getDefaultCipherSuites ()`
[pure virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

`getSupportedCipherSuites()` (p.3347)

Implemented in `decaf::internal::net::ssl::DefaultSSLSocketFactory` (p.1592), and `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory` (p.2695).

6.762.3.4 `virtual std::vector<std::string> decaf::net::ssl::SSLSocketFactory::getSupportedCipherSuites ()`
[pure virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

`getDefaultCipherSuites()` (p. 3347)

Implemented in `decaf::internal::net::ssl::DefaultSSLSocketFactory` (p. 1592), and `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory` (p. 2695).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLSocketFactory.h`

6.763 activemq::transport::tcp::SslTransport Class Reference

Transport (p. 3629) for connecting to a Broker using an SSL Socket.

`#include <src/main/activemq/transport/tcp/SslTransport.h>`

Inheritance diagram for `activemq::transport::tcp::SslTransport`:

Public Member Functions

- **SslTransport** (const **Pointer**< **Transport** > &next)
*Creates a new instance of the **SslTransport** (p. 3347), the transport will not attempt to connect to a remote host until the connect method is called.*
- virtual **~SslTransport** ()

Protected Member Functions

- virtual `decaf::net::Socket * createSocket ()`
Create an unconnected Socket instance to be used by the transport to communicate with the broker.
Returns
a newly created unconnected Socket instance.
Exceptions
IOException if there is an error while creating the unconnected Socket.
- virtual void **configureSocket** (decaf::net::Socket *socket, decaf::util::Properties &properties)

6.763.1 Detailed Description

Transport (p. 3629) for connecting to a Broker using an SSL Socket. This transport simply wraps the **TcpTransport** (p. 3510) and provides the **TcpTransport** (p. 3510) an SSL based Socket pointer allowing the core **TcpTransport** (p. 3510) logic to be reused.

Since

3.2.0

6.763.2 Constructor & Destructor Documentation

6.763.2.1 `activemq::transport::tcp::SslTransport::SslTransport (const Pointer< Transport > & next)`

Creates a new instance of the **SslTransport** (p. 3347), the transport will not attempt to connect to a remote host until the connect method is called.

Parameters

next the next transport in the chain

6.763.2.2 `virtual activemq::transport::tcp::SslTransport::~~SslTransport ()` [virtual]

6.763.3 Member Function Documentation

6.763.3.1 `virtual void activemq::transport::tcp::SslTransport::configureSocket (decaf::net::Socket * socket, decaf::util::Properties & properties)` [protected, virtual]

6.763.3.2 `virtual decaf::net::Socket* activemq::transport::tcp::SslTransport::createSocket ()` [protected, virtual]

Create an unconnected Socket instance to be used by the transport to communicate with the broker.

Returns

a newly created unconnected Socket instance.

Exceptions

IOException if there is an error while creating the unconnected Socket.

Reimplemented from **activemq::transport::tcp::TcpTransport** (p. 3513).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/SslTransport.h`

6.764 activemq::transport::tcp::SslTransportFactory Class Reference

```
#include <src/main/activemq/transport/tcp/SslTransportFactory.h>
```

Inheritance diagram for activemq::transport::tcp::SslTransportFactory:

Public Member Functions

- virtual `~SslTransportFactory()`

Protected Member Functions

- virtual `Pointer< Transport > doCreateComposite (const decaf::net::URI &location, const Pointer< wireformat::WireFormat > &wireFormat, const decaf::util::Properties &properties) throw (exceptions::ActiveMQException)`

*Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.*

6.764.1 Constructor & Destructor Documentation

- 6.764.1.1 virtual
`activemq::transport::tcp::SslTransportFactory::~~SslTransportFactory () [virtual]`

6.764.2 Member Function Documentation

- 6.764.2.1 virtual `Pointer<Transport> activemq::transport::tcp::SslTransportFactory::doCreateComposite (const decaf::net::URI & location, const Pointer< wireformat::WireFormat > & wireFormat, const decaf::util::Properties & properties) throw (exceptions::ActiveMQException) [protected, virtual]`

Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to.

wireFormat - the assigned WireFormat for the new **Transport** (p. 3629).

properties - Properties to apply to the transport.

Returns

new Pointer to a **SslTransport** (p. 3347).

Exceptions

ActiveMQException if an error occurs

Reimplemented from **activemq::transport::tcp::TcpTransportFactory** (p. 3515).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/SslTransportFactory.h`

6.765 **activemq::commands::BrokerError::StackTraceElement** Struct Reference

```
#include <src/main/activemq/commands/BrokerError.h>
```

Data Fields

- `std::string` **ClassName**
- `std::string` **FileName**
- `std::string` **MethodName**
- `int` **LineNumber**

6.765.1 Field Documentation

6.765.1.1 `std::string` **activemq::commands::BrokerError::StackTraceElement::ClassName**

6.765.1.2 `std::string` **activemq::commands::BrokerError::StackTraceElement::FileName**

6.765.1.3 `int` **activemq::commands::BrokerError::StackTraceElement::LineNumber**

6.765.1.4 `std::string` **activemq::commands::BrokerError::StackTraceElement::MethodName**

The documentation for this struct was generated from the following file:

- `src/main/activemq/commands/BrokerError.h`

6.766 **decaf::internal::io::StandardErrorOutputStream** Class Reference

Wrapper Around the Standard error Output facility on the current platform.

```
#include <src/main/decaf/internal/io/StandardErrorOutputStream.h>
```

Inheritance diagram for `decaf::internal::io::StandardErrorOutputStream`:

Public Member Functions

- **StandardErrorOutputStream** ()

- virtual `~StandardErrorOutputStream ()`
- virtual void **flush** () throw (decaf::io::IOException)
The default implementation of this method does nothing.
- virtual void **close** () throw (decaf::io::IOException)
The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw (decaf::io::IOException)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

6.766.1 Detailed Description

Wrapper Around the Standard error Output facility on the current platform. This allows for the use of alternate output methods on platforms or compilers that do not support `std::cerr`.

6.766.2 Constructor & Destructor Documentation

- 6.766.2.1** decaf::internal::io::StandardErrorOutputStream::StandardErrorOutputStream ()
- 6.766.2.2** virtual decaf::internal::io::StandardErrorOutputStream::~~StandardErrorOutputStream () [virtual]

6.766.3 Member Function Documentation

- 6.766.3.1** virtual void decaf::internal::io::StandardErrorOutputStream::close () throw (decaf::io::IOException) [virtual]

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from `decaf::io::OutputStream` (p. 2720).

- 6.766.3.2** virtual void decaf::internal::io::StandardErrorOutputStream::doWriteArrayBounded (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [protected, virtual]

Reimplemented from `decaf::io::OutputStream` (p. 2721).

6.766.3.3 virtual void decaf::internal::io::StandardErrorOutputStream::doWriteByte (unsigned char *value*) throw (decaf::io::IOException) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2721).

6.766.3.4 virtual void decaf::internal::io::StandardErrorOutputStream::flush () throw (decaf::io::IOException) [virtual]

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2721).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/StandardErrorOutputStream.h

6.767 decaf::internal::io::StandardInputStream Class Reference

```
#include <src/main/decaf/internal/io/StandardInputStream.h>
```

Inheritance diagram for decaf::internal::io::StandardInputStream:

Public Member Functions

- **StandardInputStream** ()
- virtual **~StandardInputStream** ()
- virtual int **available** () const throw (decaf::io::IOException)

Indicates the number of bytes available.

Protected Member Functions

- virtual int **doReadByte** () throw (decaf::io::IOException)

6.767.1 Constructor & Destructor Documentation

6.767.1.1 decaf::internal::io::StandardInputStream::StandardInputStream ()

6.767.1.2 virtual decaf::internal::io::StandardInputStream::~~StandardInputStream () [virtual]

6.767.2 Member Function Documentation

6.767.2.1 virtual int decaf::internal::io::StandardInputStream::available () const throw (decaf::io::IOException) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1911).

6.767.2.2 virtual int decaf::internal::io::StandardInputStream::doReadByte () throw (decaf::io::IOException) [protected, virtual]

Implements **decaf::io::InputStream** (p. 1912).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/StandardInputStream.h

6.768 decaf::internal::io::StandardOutputStream Class Reference

```
#include <src/main/decaf/internal/io/StandardOutputStream.h>
```

Inheritance diagram for decaf::internal::io::StandardOutputStream:

Public Member Functions

- **StandardOutputStream** ()
- virtual **~StandardOutputStream** ()

- virtual void **flush** () throw (decaf::io::IOException)

The default implementation of this method does nothing.

- virtual void **close** () throw (decaf::io::IOException)

The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw (decaf::io::IOException)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

6.768.1 Constructor & Destructor Documentation

6.768.1.1 decaf::internal::io::StandardOutputStream::StandardOutputStream ()

6.768.1.2 virtual
decaf::internal::io::StandardOutputStream::~~StandardOutputStream () [virtual]

6.768.2 Member Function Documentation

6.768.2.1 virtual void decaf::internal::io::StandardOutputStream::close () throw (decaf::io::IOException) [virtual]

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2720).

6.768.2.2 virtual void decaf::internal::io::StandardOutputStream::doWriteArrayBounded (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2721).

6.768.2.3 virtual void decaf::internal::io::StandardOutputStream::doWriteByte (unsigned char *value*) throw (decaf::io::IOException) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2721).

6.768.2.4 virtual void decaf::internal::io::StandardOutputStream::flush () throw (decaf::io::IOException) [virtual]

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2721).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardOutputStream.h**

6.769 cms::Startable Class Reference

Interface for a class that implements the start method.

```
#include <src/main/cms/Startable.h>
```

Inheritance diagram for cms::Startable:

Public Member Functions

- virtual ~**Startable** ()
- virtual void **start** ()=0 throw (CMSEException)

Starts the service.

6.769.1 Detailed Description

Interface for a class that implements the start method. An object that implements the **Startable** (p. 3355) interface implies that until its start method is called it will be considered to be in a closed or stopped state and will throw an Exception to indicate that it is not in an started state if one of its methods is called.

Since

1.0

6.769.2 Constructor & Destructor Documentation

6.769.2.1 virtual cms::Startable::~~Startable () [inline, virtual]

6.769.3 Member Function Documentation

6.769.3.1 virtual void cms::Startable::start () throw (CMSEException) [pure virtual]

Starts the service.

Exceptions

CMSException (p. 1074) if an internal error occurs while starting.

The documentation for this class was generated from the following file:

- `src/main/cms/Startable.h`

6.770 decaf::lang::STATIC_CAST_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.771 activemq::core::ActiveMQConstants::StaticInitializer Class Reference

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

Public Member Functions

- `StaticInitializer ()`
- `virtual ~StaticInitializer ()`

Static Public Attributes

- `static std::string destOptions [NUM_OPTIONS]`
- `static std::string uriParams [NUM_PARAMS]`
- `static std::map< std::string, DestinationOption > destOptionMap`
- `static std::map< std::string, URIParam > uriParamsMap`

6.771.1 Constructor & Destructor Documentation

6.771.1.1 `activemq::core::ActiveMQConstants::StaticInitializer::StaticInitializer ()`

6.771.1.2 `virtual
activemq::core::ActiveMQConstants::StaticInitializer::~~StaticInitializer ()` [inline, virtual]

6.771.2 Field Documentation

6.771.2.1 `std::map<std::string, DestinationOption> activemq::core::ActiveMQConstants::StaticInitializer::destOptionMap` [static]

6.771.2.2 `std::string
activemq::core::ActiveMQConstants::StaticInitializer::destOptions[NUM_OPTIONS]` [static]

6.771.2.3 `std::string
activemq::core::ActiveMQConstants::StaticInitializer::uriParams[NUM_PARAMS]` [static]

6.771.2.4 `std::map<std::string, URIParam> activemq::core::ActiveMQConstants::StaticInitializer::uriParamsMap` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConstants.h`

6.772 decaf::util::StlList< E > Class Template Reference

List (p. 2190) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

```
#include <src/main/decaf/util/StlList.h>
```

Inheritance diagram for `decaf::util::StlList< E >`:

Data Structures

- class `ConstStlListIterator`
- class `StlListIterator`

Public Member Functions

- `StlList ()`
Default constructor - does nothing.

- **StlList** (const **StlList** &source)

Copy constructor - copies the content of the given set into this one.

- **StlList** (const **Collection**< E > &source)

Copy constructor - copies the content of the given set into this one.

- virtual ~**StlList** ()
- virtual bool **equals** (const **StlList** &source) const

- virtual **Iterator**< E > * **iterator** ()

- virtual **Iterator**< E > * **iterator** () const
- virtual **ListIterator**< E > * **listIterator** ()

Returns

a list iterator over the elements in this list (in proper sequence).

- virtual **ListIterator**< E > * **listIterator** () const
- virtual **ListIterator**< E > * **listIterator** (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Parameters

index index of first element to be returned from the list iterator (by a call to the next method).

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

***IndexOutOfBoundsException** if the index is out of range ($index < 0 \parallel index > size()$ (p. 1106))*

- virtual **ListIterator**< E > * **listIterator** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual void **copy** (const **StlList** &source)

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

*This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2014) operation. Most implementations will probably choose to override this method for efficiency.*

*Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*

Exceptions

***UnsupportedOperationException** if the clear operation is not supported by this collection*

- virtual bool **contains** (const E &value) const throw (lang::Exception)

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters

value - the value whose presence is to be queried for in this **Collection** (p. 1097).

Returns

true if the value is contained in this collection

Exceptions

Exception if an error occurs,

- virtual std::size_t **indexOf** (const E &value) throw (decaf::lang::exceptions::NoSuchElementException)

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index i such that $get(i) == value$, or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NoSuchElementException if value is not in the list

- virtual std::size_t **lastIndexOf** (const E &value) throw (decaf::lang::exceptions::NoSuchElementException)

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index i such that $get(i) == value$ or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

NoSuchElementException if value is not in the list

- virtual bool **isEmpty** () const

Returns true if this collection contains no elements.

*This implementation returns **size()** (p. 1106) == 0.*

Returns

true if the size method return 0.

- virtual std::size_t **size** () const

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX_VALUE elements, returns Integer.MAX_VALUE.

Returns

the number of elements in this collection

- virtual E **get** (std::size_t index) const throw (default::lang::exceptions::IndexOutOfBoundsException)

Gets the element contained at position passed.

Parameters

index - position to get

Returns

value at index

- virtual E **set** (std::size_t index, const E &element) throw (default::lang::exceptions::IndexOutOfBoundsException)

Replaces the element at the specified position in this list with the specified element.

Parameters

index - index of the element to replace

element - element to be stored at the specified position

Returns

the element previously at the specified position

Exceptions

IndexOutOfBoundsException - if the index is greater than size

- virtual bool **add** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

*Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1097) classes should clearly specify in their documentation any restrictions on what elements may be added.*

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value - reference to the element to add.

Returns

true if the element was added

Exceptions

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

- virtual void **add** (std::size_t index, const E &element) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IndexOutOfBoundsException)

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

index - index at which the specified element is to be inserted

element - element to be inserted

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

- virtual bool **addAll** (std::size_t index, const **Collection**< E > &source) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

index The index at which to insert the first element from the specified collection

source The **Collection** (p. 1097) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element e such that e == o, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 1097).

- virtual E **remove** (std::size_t index) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

*Removes the element at the specified position in this list.
Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.*

Parameters

***index** - the index of the element to be removed*

Returns

the element previously at the specified position

Exceptions

***IndexOutOfBoundsException** - if the index is greater than size
UnsupportedOperationException - If the collection is non-modifiable.*

6.772.1 Detailed Description

```
template<typename E> class decaf::util::StlList< E >
```

List (p. 2190) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

6.772.2 Constructor & Destructor Documentation

6.772.2.1 `template<typename E> decaf::util::StlList< E >::StlList () [inline]`

Default constructor - does nothing.

6.772.2.2 `template<typename E> decaf::util::StlList< E >::StlList (const StlList< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

Parameters

source The source set.

6.772.2.3 `template<typename E> decaf::util::StlList< E >::StlList (const Collection< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

Parameters

source The source set.

6.772.2.4 `template<typename E> virtual decaf::util::StlList< E >::~~StlList ()`
`[inline, virtual]`

6.772.3 Member Function Documentation

6.772.3.1 `template<typename E> virtual bool decaf::util::StlList< E >::add (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p.1097) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value - reference to the element to add.

Returns

true if the element was added

Exceptions

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p.1099).

Referenced by `decaf::util::StlList< cms::Connection * >::addAll()`.

6.772.3.2 `template<typename E> virtual void decaf::util::StlList< E >::add (std::size_t index, const E & element) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

index - index at which the specified element is to be inserted

element - element to be inserted

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 2192).

```
6.772.3.3  template<typename E> virtual bool decaf::util::StlList< E >::addAll
            ( std::size_t index, const Collection< E > & source )
            throw ( decaf::lang::exceptions::UnsupportedOperationException,
                    decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,
                                virtual]
```

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

index The index at which to insert the first element from the specified collection

source The **Collection** (p. 1097) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 2192).

```
6.772.3.4  template<typename E> virtual void decaf::util::StlList< E >::clear ( )
            throw ( lang::exceptions::UnsupportedOperationException ) [inline,
                                virtual]
```

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2014) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

UnsupportedOperationException if the clear operation is not supported by this collection

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 147).

6.772.3.5 `template<typename E> virtual bool decaf::util::StlList< E >::contains (const E & value) const throw (lang::Exception) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters

value - the value whose presence is to be queried for in this **Collection** (p. 1097).

Returns

true if the value is contained in this collection

Exceptions

Exception if an error occurs,

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 148).

6.772.3.6 `template<typename E> virtual void decaf::util::StlList< E >::copy (const StlList< E > & source) [inline, virtual]`

Referenced by `decaf::util::StlList< cms::Connection * >::StlList()`.

6.772.3.7 `template<typename E> virtual bool decaf::util::StlList< E >::equals (const StlList< E > & source) const [inline, virtual]`

6.772.3.8 `template<typename E> virtual E decaf::util::StlList< E >::get (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Gets the element contained at position passed.

Parameters

index - position to get

Returns

value at index

Implements `decaf::util::List< E >` (p. 2193).

6.772.3.9 `template<typename E> virtual std::size_t decaf::util::StlList< E >::indexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NoSuchElementException if value is not in the list

Implements `decaf::util::List< E >` (p. 2193).

6.772.3.10 `template<typename E> virtual bool decaf::util::StlList< E >::isEmpty () const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns `size() (p. 1106) == 0`.

Returns

true if the size method return 0.

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 150).

6.772.3.11 `template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator () [inline, virtual]`

6.772.3.12 `template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator () const [inline, virtual]`

6.772.3.13 `template<typename E> virtual std::size_t decaf::util::StlList< E >::lastIndexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

NoSuchElementException if value is not in the list

Implements **decaf::util::List< E >** (p. 2194).

6.772.3.14 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Parameters

index index of first element to be returned from the list iterator (by a call to the next method).

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

IndexOutOfBoundsException if the index is out of range (`index < 0 || index > size()` (p. 1106))

Implements **decaf::util::List< E >** (p. 2195).

6.772.3.15 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator () [inline, virtual]`

Returns

a list iterator over the elements in this list (in proper sequence).

Implements **decaf::util::List< E >** (p. 2195).

6.772.3.16 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator () const [inline, virtual]`

Implements **decaf::util::List< E >** (p. 2195).

6.772.3.17 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Implements **decaf::util::List< E >** (p. 2195).

6.772.3.18 `template<typename E> virtual E decaf::util::StlList< E >::remove (std::size_t index) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

index - the index of the element to be removed

Returns

the element previously at the specified position

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implements `decaf::util::List< E >` (p. 2196).

6.772.3.19 `template<typename E> virtual bool decaf::util::StlList< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 1097).

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 151).

6.772.3.20 `template<typename E> virtual E decaf::util::StlList< E >::set (std::size_t index, const E & element) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Replaces the element at the specified position in this list with the specified element.

Parameters

index - index of the element to replace

element - element to be stored at the specified position

Returns

the element previously at the specified position

Exceptions

IndexOutOfBoundsException - if the index is greater than size

Implements `decaf::util::List< E >` (p. 2196).

6.772.3.21 `template<typename E> virtual std::size_t decaf::util::StlList< E >::size () const [inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX_VALUE elements, returns Integer.MAX_VALUE.

Returns

the number of elements in this collection

Implements `decaf::util::Collection< E >` (p. 1106).

Referenced by `decaf::util::StlList< cms::Connection * >::add()`, `decaf::util::StlList< cms::Connection * >::addAll()`, `decaf::util::StlList< cms::Connection * >::get()`, `decaf::util::StlList< cms::Connection * >::lastIndexOf()`, `decaf::util::StlList< cms::Connection * >::listIterator()`, `decaf::util::StlList< cms::Connection * >::remove()`, and `decaf::util::StlList< cms::Connection * >::set()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlList.h`

6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference

Map (p. 2305) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

`#include <src/main/decaf/util/StlMap.h>`

Inheritance diagram for `decaf::util::StlMap< K, V, COMPARATOR >`:

Public Member Functions

- **StlMap** ()
Default constructor - does nothing.
- **StlMap** (const **StlMap** &source)
Copy constructor - copies the content of the given map into this one.
- **StlMap** (const **Map**< K, V, COMPARATOR > &source)
Copy constructor - copies the content of the given map into this one.
- virtual ~**StlMap** ()
- virtual bool **equals** (const **StlMap** &source) const
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const
Comparison, equality is dependent on the method of determining if the element are equal.
Parameters
source - **Map** (p. 2305) to compare to this one.
Returns
*true if the **Map** (p. 2305) passed is equal in value to this one.*
- virtual void **copy** (const **StlMap** &source)
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)
Copies the content of the source map into this map.
Erases all existing data in this map.
Parameters
source The source object to copy from.
- virtual void **clear** () throw (decaf::lang::exceptions::UnsupportedOperationException)
Removes all keys and values from this map.
Exceptions
UnsupportedOperationException if this map is unmodifiable.
- virtual bool **containsKey** (const K &key) const
Indicates whether or this map contains a value for the given key.
Parameters
key The key to look up.
Returns
true if this map contains the value, otherwise false.
- virtual bool **containsValue** (const V &value) const
Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.
Parameters
value The Value to look up.

Returns

true if this map contains the value, otherwise false.

- virtual bool **isEmpty** () const

Returns

*if the **Map** (p. 2305) contains any element or not, TRUE or FALSE*

- virtual std::size_t **size** () const

Returns

The number of elements (key/value pairs) in this map.

- virtual V & **get** (const K &key) throw (lang::exceptions::NoSuchElementException)

*Gets the value mapped to the specified key in the **Map** (p. 2305).*

*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*

Parameters

key The search key.

Returns

A reference to the value for the given key.

Exceptions

***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2305).*

- virtual const V & **get** (const K &key) const throw (lang::exceptions::NoSuchElementException)

*Gets the value mapped to the specified key in the **Map** (p. 2305).*

*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*

Parameters

key The search key.

Returns

A {const} reference to the value for the given key.

Exceptions

***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2305).*

- virtual void **put** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)

Sets the value for the specified key.

Parameters

key The target key.

value The value to be set.

Exceptions

***UnsupportedOperationException** if this map is unmodifiable.*

- virtual void **putAll** (const StlMap< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

- virtual void **putAll** (const Map< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

*Stores a copy of the Mappings contained in the other **Map** (p. 2305) in this one.*

Parameters

*other A **Map** (p. 2305) instance whose elements are to all be inserted in this **Map** (p. 2305).*

Exceptions

UnsupportedOperationException *If the implementing class does not support the putAll operation.*

- virtual V **remove** (const K &key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

key The search key.

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException *if this key is not in the **Map** (p. 2305).*
UnsupportedOperationException *if this map is unmodifiable.*

- virtual std::vector< K > **keySet** () const

*Returns a **Set** (p. 3220) view of the mappings contained in this map.*

*The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2014), **Set.remove** (p. 151), removeAll, retainAll and clear operations. It does not support the add or addAll operations.*

Returns

the entire set of keys in this map as a std::vector.

- virtual std::vector< V > **values** () const

Returns

the entire set of values in this map as a std::vector.

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)

Unlocks the object.

- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.773.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::StlMap< K, V, COMPARATOR >
```

Map (p. 2305) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

Since

1.0

6.773.2 Constructor & Destructor Documentation

6.773.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap () [inline]`

Default constructor - does nothing.

6.773.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap (const StlMap< K, V, COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters

source The source map.

6.773.2.3 `template<typename K, typename V, typename COMPARTOR =
std::less<K>> decaf::util::StlMap< K, V, COMPARTOR >::StlMap (`
`const Map< K, V, COMPARTOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters

source The source map.

6.773.2.4 `template<typename K, typename V, typename COMPARTOR =
std::less<K>> virtual decaf::util::StlMap< K, V, COMPARTOR
>::~StlMap () [inline, virtual]`

6.773.3 Member Function Documentation

6.773.3.1 `template<typename K, typename V, typename COMPARTOR
= std::less<K>> virtual void decaf::util::StlMap<
K, V, COMPARTOR >::clear () throw (
decaf::lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Removes all keys and values from this map.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implements `decaf::util::Map< K, V, COMPARTOR >` (p. 2307).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::copy()`.

6.773.3.2 `template<typename K, typename V, typename COMPARTOR =
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARTOR
>::containsKey (const K & key) const [inline, virtual]`

Indicates whether or this map contains a value for the given key.

Parameters

key The key to look up.

Returns

true if this map contains the value, otherwise false.

Implements `decaf::util::Map< K, V, COMPARTOR >` (p. 2308).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::equals()`.

6.773.3.3 `template<typename K, typename V, typename COMPARTOR =
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARTOR
>::containsValue (const V & value) const [inline, virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters

value The Value to look up.

Returns

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2309).

6.773.3.4 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::copy (const StlMap< K, V, COMPARATOR > & source)
[inline, virtual]`

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::StlMap()`.

6.773.3.5 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::copy (const Map< K, V, COMPARATOR > & source) [inline,
virtual]`

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters

source The source object to copy from.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2309).

6.773.3.6 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR
>::equals (const Map< K, V, COMPARATOR > & source) const
[inline, virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters

source - **Map** (p. 2305) to compare to this one.

Returns

true if the **Map** (p. 2305) passed is equal in value to this one.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2310).

6.773.3.7 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR
>::equals (const StlMap< K, V, COMPARATOR > & source) const
[inline, virtual]`

6.773.3.8 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual V& decaf::util::StlMap< K, V,
COMPARATOR >::get (const K & key) throw (
lang::exceptions::NoSuchElementException) [inline, virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2305).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters

key The search key.

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 2305).

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2310).

6.773.3.9 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual const V& decaf::util::StlMap< K, V,
COMPARATOR >::get (const K & key) const throw (
lang::exceptions::NoSuchElementException) [inline, virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2305).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters

key The search key.

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 2305).

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2311).

6.773.3.10 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::isEmpty () const [inline, virtual]`

Returns

if the **Map** (p. 2305) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2312).

6.773.3.11 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::vector<K> decaf::util::StlMap< K, V, COMPARATOR >::keySet () const [inline, virtual]`

Returns a **Set** (p. 3220) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2014), **Set.remove** (p. 151), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

Returns

the entire set of keys in this map as a `std::vector`.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2313).

6.773.3.12 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3463).

6.773.3.13 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3464).

```
6.773.3.14  template<typename K, typename V, typename COMPARATOR =
            std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
            >::notifyAll (    ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
            virtual]
```

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3465).

```
6.773.3.15  template<typename K, typename V, typename COMPARATOR =
            std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
            >::put ( const K & key, const V & value ) throw (
            decaf::lang::exceptions::UnsupportedOperationException ) [inline,
            virtual]
```

Sets the value for the specified key.

Parameters

key The target key.

value The value to be set.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2314).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::putAll()`.

```
6.773.3.16  template<typename K, typename V, typename COMPARATOR =
            std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
            >::putAll ( const StlMap< K, V, COMPARATOR > & other ) throw
            ( decaf::lang::exceptions::UnsupportedOperationException ) [inline,
            virtual]
```

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::copy()`.

6.773.3.17 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & other) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 2305) in this one.

Parameters

other A **Map** (p. 2305) instance whose elements are to all be inserted in this **Map** (p. 2305).

Exceptions

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2314).

6.773.3.18 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::StlMap< K, V, COMPARATOR >::remove (const K & key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

key The search key.

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException if this key is not in the **Map** (p. 2305).

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2315).

6.773.3.19 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::size_t decaf::util::StlMap< K, V, COMPARATOR >::size () const [inline, virtual]`

Returns

The number of elements (key/value pairs) in this map.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2316).

6.773.3.20 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR
>::tryLock () throw (decaf::lang::exceptions::RuntimeException)
[inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3466).

6.773.3.21 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::unlock () throw (decaf::lang::exceptions::RuntimeException)
[inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3467).

6.773.3.22 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual std::vector<V> decaf::util::StlMap< K, V,
COMPARATOR >::values () const [inline, virtual]`

Returns

the entire set of values in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2317).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::values()`.

6.773.3.23 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::wait () throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3468).

```
6.773.3.24  template<typename K, typename V, typename COMPARATOR
            = std::less<K>> virtual void decaf::util::StlMap< K, V,
            COMPARATOR >::wait ( long long millisecs, int
            nanos ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalArgumentException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3471).

```
6.773.3.25  template<typename K, typename V, typename COMPARATOR
            = std::less<K>> virtual void decaf::util::StlMap< K,
            V, COMPARATOR >::wait ( long long millisecs )
            throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3470).

The documentation for this class was generated from the following file:

- src/main/decaf/util/StlMap.h

6.774 decaf::util::StlQueue< T > Class Template Reference

The **Queue** (p. 2948) class accepts messages with an psuh(m) command where m is the message to be queued.

```
#include <src/main/decaf/util/StlQueue.h>
```

Inheritance diagram for decaf::util::StlQueue< T >:

Data Structures

- class **QueueIterator**

Public Member Functions

- **StlQueue** ()
- virtual **~StlQueue** ()
- **Iterator**< T > * **iterator** ()
*Gets an **Iterator** (p. 2012) over this **Queue** (p. 2948).*
- void **clear** ()
Empties this queue.
- T & **front** ()
Returns a Reference to the element at the head of the queue.
- const T & **front** () const
Returns a Reference to the element at the head of the queue.
- T & **back** ()
Returns a Reference to the element at the tail of the queue.

- `const T & back () const`
Returns a Reference to the element at the tail of the queue.
- `void push (const T &t)`
Places a new Object at the Tail of the queue.
- `void enqueueFront (const T &t)`
Places a new Object at the front of the queue.
- `T pop ()`
Removes and returns the element that is at the Head of the queue.
- `size_t size () const`
*Gets the Number of elements currently in the **Queue** (p. 2948).*
- `bool empty () const`
*Checks if this **Queue** (p. 2948) is currently empty.*
- `virtual std::vector< T > toArray () const`
- `void reverse (StlQueue< T > &target) const`
Reverses the order of the contents of this queue and stores them in the target queue.
- `virtual void lock () throw (decaf::lang::exceptions::RuntimeException)`
Locks the object.
- `virtual bool tryLock () throw (decaf::lang::exceptions::RuntimeException)`
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- `virtual void unlock () throw (decaf::lang::exceptions::RuntimeException)`
Unlocks the object.
- `virtual void wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)`
Waits on a signal from this object, which is generated by a call to Notify.
- `virtual void wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)`
Waits on a signal from this object, which is generated by a call to Notify.
- `virtual void wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)`
Waits on a signal from this object, which is generated by a call to Notify.
- `virtual void notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)`

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

- T & **getSafeValue** ()

Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

6.774.1 Detailed Description

template<typename T> class decaf::util::StlQueue< T >

The **Queue** (p. 2948) class accepts messages with an psuh(m) command where m is the message to be queued. It destructively returns the message with **pop()** (p. 3387). **pop()** (p. 3387) returns messages in the order they were enqueued.

Queue (p. 2948) is implemented with an instance of the STL queue object. The interface is essentially the same as that of the STL queue except that the pop method actually reaturns a reference to the element popped. This frees the app from having to call the **front** method before calling pop.

```
Queue<string> sq; // make a queue to hold string messages sq.push(s); // enqueues a message
m string s = sq.pop(); // dequeues a message
```

= DESIGN CONSIDERATIONS

The **Queue** (p. 2948) class inherits from the Synchronizable interface and provides methods for locking and unlocking this queue as well as waiting on this queue. In a multi-threaded app this can allow for multiple threads to be reading from and writing to the same **Queue** (p. 2948).

Clients should consider that in a multiple threaded app it is possible that items could be placed on the queue faster than you are taking them off, so protection should be placed in your polling loop to ensure that you don't get stuck there.

6.774.2 Constructor & Destructor Documentation

6.774.2.1 **template<typename T> decaf::util::StlQueue< T >::StlQueue ()**
[inline]

6.774.2.2 **template<typename T> virtual decaf::util::StlQueue< T >::~~StlQueue ()**
[inline, virtual]

6.774.3 Member Function Documentation

6.774.3.1 **template<typename T> T& decaf::util::StlQueue< T >::back ()**
[inline]

Returns a Reference to the element at the tail of the queue.

Returns

reference to a queue type object or (safe)

6.774.3.2 `template<typename T> const T& decaf::util::StlQueue< T >::back ()`
`const [inline]`

Returns a Reference to the element at the tail of the queue.

Returns

reference to a queue type object or (safe)

6.774.3.3 `template<typename T> void decaf::util::StlQueue< T >::clear ()`
`[inline]`

Empties this queue.

6.774.3.4 `template<typename T> bool decaf::util::StlQueue< T >::empty ()`
`const [inline]`

Checks if this **Queue** (p. 2948) is currently empty.

Returns

boolean indicating queue emptiness

6.774.3.5 `template<typename T> void decaf::util::StlQueue< T >::enqueueFront (`
`const T & t) [inline]`

Places a new Object at the front of the queue.

Parameters

t - **Queue** (p. 2948) Object Type reference.

6.774.3.6 `template<typename T> const T& decaf::util::StlQueue< T >::front ()`
`const [inline]`

Returns a Reference to the element at the head of the queue.

Returns

reference to a queue type object or (safe)

6.774.3.7 `template<typename T> T& decaf::util::StlQueue< T >::front ()`
`[inline]`

Returns a Reference to the element at the head of the queue.

Returns

reference to a queue type object or (safe)

6.774.3.8 `template<typename T> T& decaf::util::StlQueue< T >::getSafeValue () [inline]`

Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

Returns

Reference to this Queues safe object

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::back()`, `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::front()`, and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::pop()`.

6.774.3.9 `template<typename T> Iterator<T>* decaf::util::StlQueue< T >::iterator () [inline]`

Gets an **Iterator** (p. 2012) over this **Queue** (p. 2948).

Returns

new iterator pointer that is owned by the caller.

6.774.3.10 `template<typename T> virtual void decaf::util::StlQueue< T >::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3463).

6.774.3.11 `template<typename T> virtual void decaf::util::StlQueue< T >::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3464).

6.774.3.12 `template<typename T> virtual void decaf::util::StlQueue< T >::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3465).

6.774.3.13 `template<typename T> T decaf::util::StlQueue< T >::pop () [inline]`

Removes and returns the element that is at the Head of the queue.

Returns

reference to a queue type object or (safe)

6.774.3.14 `template<typename T> void decaf::util::StlQueue< T >::push (const T & t) [inline]`

Places a new Object at the Tail of the queue.

Parameters

t - **Queue** (p. 2948) Object Type reference.

6.774.3.15 `template<typename T> void decaf::util::StlQueue< T >::reverse (StlQueue< T > & target) const [inline]`

Reverses the order of the contents of this queue and stores them in the target queue.

Parameters

target - The target queue that will receive the contents of this queue in reverse order.

6.774.3.16 `template<typename T> size_t decaf::util::StlQueue< T >::size () const [inline]`

Gets the Number of elements currently in the **Queue** (p. 2948).

Returns

Queue (p. 2948) Size

6.774.3.17 `template<typename T> virtual std::vector<T> decaf::util::StlQueue< T >::toArray () const [inline, virtual]`

Returns

the all values in this queue as a std::vector.

6.774.3.18 `template<typename T> virtual bool decaf::util::StlQueue< T >::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3466).

6.774.3.19 `template<typename T> virtual void decaf::util::StlQueue< T >::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3467).

6.774.3.20 `template<typename T> virtual void decaf::util::StlQueue< T >::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3468).

6.774.3.21 `template<typename T> virtual void decaf::util::StlQueue< T >::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3470).

6.774.3.22 `template<typename T> virtual void decaf::util::StlQueue< T >::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentOutOfRangeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentOutOfRangeException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3471).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlQueue.h`

6.775 decaf::util::StlSet< E > Class Template Reference

Set (p. 3220) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.

```
#include <src/main/decaf/util/StlSet.h>
```

Inheritance diagram for `decaf::util::StlSet< E >`:

Data Structures

- class **ConstSetIterator**
- class **SetIterator**

Public Member Functions

- **StlSet** ()
Default constructor - does nothing.
- **StlSet** (const **StlSet** &source)
Copy constructor - copies the content of the given set into this one.
- **StlSet** (const **Collection**< E > &source)
Copy constructor - copies the content of the given set into this one.
- virtual **~StlSet** ()
- **Iterator**< E > * **iterator** ()
- **Iterator**< E > * **iterator** () const
- virtual bool **equals** (const **StlSet** &source) const
- virtual void **copy** (const **StlSet** &source)
- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
*Removes all of the elements from this collection (optional operation).
The collection will be empty after this method returns.
This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2014) operation. Most implementations will probably choose to override this method for efficiency.
Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*
Exceptions
***UnsupportedOperationException** if the clear operation is not supported by this collection*

- virtual bool **contains** (const E &value) const throw (lang::Exception)

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters

value - the value whose presence is to be queried for in this **Collection** (p. 1097).

Returns

true if the value is contained in this collection

Exceptions

Exception if an error occurs,

- virtual bool **isEmpty** () const
- virtual std::size_t **size** () const
- virtual bool **add** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

*Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1097) classes should clearly specify in their documentation any restrictions on what elements may be added.*

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value - reference to the element to add.

Returns

true if the element was added

Exceptions

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element e such that e == o, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 1097).

6.775.1 Detailed Description

```
template<typename E> class decaf::util::StlSet< E >
```

Set (p. 3220) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.

6.775.2 Constructor & Destructor Documentation

```
6.775.2.1  template<typename E> decaf::util::StlSet< E >::StlSet (    ) [inline]
```

Default constructor - does nothing.

```
6.775.2.2  template<typename E> decaf::util::StlSet< E >::StlSet ( const StlSet<
E > & source ) [inline]
```

Copy constructor - copies the content of the given set into this one.

Parameters

source The source set.

```
6.775.2.3  template<typename E> decaf::util::StlSet< E >::StlSet ( const
Collection< E > & source ) [inline]
```

Copy constructor - copies the content of the given set into this one.

Parameters

source The source set.

```
6.775.2.4  template<typename E> virtual decaf::util::StlSet< E >::~~StlSet (    )
[inline, virtual]
```

6.775.3 Member Function Documentation

```
6.775.3.1  template<typename E> virtual bool decaf::util::StlSet< E >::add ( const
E & value ) throw ( lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException,
lang::exceptions::IllegalStateException ) [inline,
virtual]
```

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p.1097) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value - reference to the element to add.

Returns

true if the element was added

Exceptions

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p.1099).

6.775.3.2 `template<typename E> virtual void decaf::util::StlSet< E >::clear ()
throw (lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p.2014) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

UnsupportedOperationException if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractCollection< E >** (p.147).

6.775.3.3 `template<typename E> virtual bool decaf::util::StlSet< E >::contains (const E & value) const throw (lang::Exception) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters

value - the value whose presence is to be queried for in this **Collection** (p.1097).

Returns

true if the value is contained in this collection

Exceptions

Exception if an error occurs,

Reimplemented from `decaf::util::AbstractCollection< E >` (p.148).

6.775.3.4 `template<typename E> virtual void decaf::util::StlSet< E >::copy (const StlSet< E > & source) [inline, virtual]`

Referenced by `decaf::util::StlSet< ActiveMQSession * >::StlSet()`.

6.775.3.5 `template<typename E> virtual bool decaf::util::StlSet< E >::equals (const StlSet< E > & source) const [inline, virtual]`

6.775.3.6 `template<typename E> virtual bool decaf::util::StlSet< E >::isEmpty () const [inline, virtual]`

Returns

if the set contains any element or not, TRUE or FALSE

Reimplemented from `decaf::util::AbstractCollection< E >` (p.150).

6.775.3.7 `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator () const [inline]`

6.775.3.8 `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator () [inline]`

6.775.3.9 `template<typename E> virtual bool decaf::util::StlSet< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 1097).

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 151).

6.775.3.10 `template<typename E> virtual std::size_t decaf::util::StlSet< E >::size () const [inline, virtual]`

Returns

The number of elements in this set.

Implements `decaf::util::Collection< E >` (p. 1106).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlSet.h`

6.776 activemq::wireformat::stomp::StompCommandConstants Class Reference

```
#include <src/main/activemq/wireformat/stomp/StompCommandConstants.h>
```

Static Public Attributes

- static const std::string **CONNECT**
- static const std::string **CONNECTED**
- static const std::string **DISCONNECT**
- static const std::string **SUBSCRIBE**
- static const std::string **UNSUBSCRIBE**
- static const std::string **MESSAGE**

- static const std::string **SEND**
- static const std::string **BEGIN**
- static const std::string **COMMIT**
- static const std::string **ABORT**
- static const std::string **ACK**
- static const std::string **ERROR_CMD**
- static const std::string **RECEIPT**
- static const std::string **HEADER_DESTINATION**
- static const std::string **HEADER_TRANSACTIONID**
- static const std::string **HEADER_CONTENTLENGTH**
- static const std::string **HEADER_SESSIONID**
- static const std::string **HEADER_RECEIPT_REQUIRED**
- static const std::string **HEADER_RECEIPTID**
- static const std::string **HEADER_MESSAGEID**
- static const std::string **HEADER_ACK**
- static const std::string **HEADER_LOGIN**
- static const std::string **HEADER_PASSWORD**
- static const std::string **HEADER_CLIENT_ID**
- static const std::string **HEADER_MESSAGE**
- static const std::string **HEADER_CORRELATIONID**
- static const std::string **HEADER_REQUESTID**
- static const std::string **HEADER_RESPONSEID**
- static const std::string **HEADER_EXPIRES**
- static const std::string **HEADER_PERSISTENT**
- static const std::string **HEADER_REPLYTO**
- static const std::string **HEADER_TYPE**
- static const std::string **HEADER_DISPATCH_ASYNC**
- static const std::string **HEADER_EXCLUSIVE**
- static const std::string **HEADER_MAXPENDINGMSGLIMIT**
- static const std::string **HEADER_NOLOCAL**
- static const std::string **HEADER_PREFETCHSIZE**
- static const std::string **HEADER_JMSPRIORITY**
- static const std::string **HEADER_CONSUMERPRIORITY**
- static const std::string **HEADER_RETROACTIVE**
- static const std::string **HEADER_SUBSCRIPTIONNAME**
- static const std::string **HEADER_OLDSUBSCRIPTIONNAME**
- static const std::string **HEADER_TIMESTAMP**
- static const std::string **HEADER_REDELIVERED**
- static const std::string **HEADER_REDELIVERYCOUNT**
- static const std::string **HEADER_SELECTOR**
- static const std::string **HEADER_ID**
- static const std::string **HEADER_SUBSCRIPTION**
- static const std::string **HEADER_TRANSFORMATION**
- static const std::string **HEADER_TRANSFORMATION_ERROR**
- static const std::string **ACK_CLIENT**
- static const std::string **ACK_AUTO**
- static const std::string **ACK_INDIVIDUAL**
- static const std::string **TEXT**
- static const std::string **BYTES**
- static const std::string **QUEUE_PREFIX**
- static const std::string **TOPIC_PREFIX**
- static const std::string **TEMPQUEUE_PREFIX**
- static const std::string **TEMPTOPIC_PREFIX**

6.776.1 Field Documentation

- 6.776.1.1** `const std::string activemq::wireformat::stomp::StompCommandConstants::ABORT`
[static]
- 6.776.1.2** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK`
[static]
- 6.776.1.3** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_AUTO`
[static]
- 6.776.1.4** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_CLIENT` [static]
- 6.776.1.5** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_INDIVIDUAL` [static]
- 6.776.1.6** `const std::string activemq::wireformat::stomp::StompCommandConstants::BEGIN`
[static]
- 6.776.1.7** `const std::string activemq::wireformat::stomp::StompCommandConstants::BYTES`
[static]
- 6.776.1.8** `const std::string activemq::wireformat::stomp::StompCommandConstants::COMMIT`
[static]
- 6.776.1.9** `const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECT`
[static]
- 6.776.1.10** `const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECTED`
[static]
- 6.776.1.11** `const std::string activemq::wireformat::stomp::StompCommandConstants::DISCONNECT`
[static]
- 6.776.1.12** `const std::string activemq::wireformat::stomp::StompCommandConstants::ERROR_CMD` [static]
- 6.776.1.13** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_ACK` [static]
- 6.776.1.14** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CLIENT_ID` [static]
- 6.776.1.15** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CONSUMERPRIORITY` [static]

- `src/main/activemq/wireformat/stomp/StompCommandConstants.h`

6.777 `activemq::wireformat::stomp::StompFrame` Class Reference

A Stomp-level message frame that encloses all messages to and from the broker.

```
#include <src/main/activemq/wireformat/stomp/StompFrame.h>
```

Public Member Functions

- **StompFrame** ()
Default constructor.
- virtual **~StompFrame** ()
Destruction.
- **StompFrame * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- void **copy** (const **StompFrame** *src)
Copies the contents of the passed Frame to this one.
- void **setCommand** (const std::string &cmd)
Sets the command for this stomp frame.
- const std::string & **getCommand** () const
Accessor for this frame's command field.
- bool **hasProperty** (const std::string &name) const
Checks if the given property is present in the Frame.
- std::string **getProperty** (const std::string &name, const std::string &fallback="") const
Gets a property from this Frame's properties and returns it, or the default value given.
- std::string **removeProperty** (const std::string &name)
Gets and remove the property specified, if the property is not set, this method returns the empty string.
- void **setProperty** (const std::string &name, const std::string &value)
Sets the property given to the value specified in this Frame's Properties.
- **decaf::util::Properties** & **getProperties** ()
Gets access to the header properties for this frame.
- const **decaf::util::Properties** & **getProperties** () const
- const std::vector< unsigned char > & **getBody** () const
Accessor for the body data of this frame.

- `std::vector< unsigned char > & getBody ()`
Non-const version of the body accessor.
- `std::size_t getBodyLength () const`
Return the number of bytes contained in this frames body.
- `void setBody (const unsigned char *bytes, std::size_t numBytes)`
Sets the body data of this frame as a byte sequence.
- `void toStream (decaf::io::DataOutputStream *stream) const throw (decaf::io::IOException)`
Writes this Frame to an OuputStream in the Stomp Wire Format.
- `void fromStream (decaf::io::DataInputStream *stream) throw (decaf::io::IOException)`
Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

6.777.1 Detailed Description

A Stomp-level message frame that encloses all messages to and from the broker.

6.777.2 Constructor & Destructor Documentation

6.777.2.1 `activemq::wireformat::stomp::StompFrame::StompFrame ()` [inline]

Default constructor.

6.777.2.2 `virtual activemq::wireformat::stomp::StompFrame::~~StompFrame ()` [inline, virtual]

Destruction.

6.777.3 Member Function Documentation

6.777.3.1 `StompFrame* activemq::wireformat::stomp::StompFrame::clone () const`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.777.3.2 `void activemq::wireformat::stomp::StompFrame::copy (const StompFrame * src)`

Copies the contents of the passed Frame to this one.

Parameters

src - Frame to copy

6.777.3.3 `void activemq::wireformat::stomp::StompFrame::fromStream (decaf::io::DataInputStream * stream) throw (decaf::io::IOException)`

Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

Parameters

stream - The stream to read the Frame from.

Exceptions

IOException if an error occurs while writing the Frame.

6.777.3.4 `const std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody () const [inline]`

Accessor for the body data of this frame.

Returns

char pointer to body data

6.777.3.5 `std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody () [inline]`

Non-const version of the body accessor.

6.777.3.6 `std::size_t activemq::wireformat::stomp::StompFrame::getBodyLength () const [inline]`

Return the number of bytes contained in this frames body.

Returns

Body bytes length.

6.777.3.7 `const std::string& activemq::wireformat::stomp::StompFrame::getCommand () const [inline]`

Accessor for this frame's command field.

6.777.3.8 `decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties () [inline]`

Gets access to the header properties for this frame.

Returns

the Properties object owned by this Frame

6.777.3.9 `const decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties () const [inline]`

6.777.3.10 `std::string activemq::wireformat::stomp::StompFrame::getProperty (const std::string & name, const std::string & fallback = "") const [inline]`

Gets a property from this Frame's properties and returns it, or the default value given.

Parameters

name - The name of the property to lookup

fallback - The default value to return if this value isn't set

Returns

string value of the property asked for.

6.777.3.11 `bool activemq::wireformat::stomp::StompFrame::hasProperty (const std::string & name) const [inline]`

Checks if the given property is present in the Frame.

Parameters

name - The name of the property to check for.

6.777.3.12 `std::string activemq::wireformat::stomp::StompFrame::removeProperty (const std::string & name) [inline]`

Gets and remove the property specified, if the property is not set, this method returns the empty string.

Parameters

name - the Name of the property to get and return.

6.777.3.13 `void activemq::wireformat::stomp::StompFrame::setBody (const unsigned char * bytes, std::size_t numBytes)`

Sets the body data of this frame as a byte sequence.

Parameters

bytes The byte buffer to be set in the body.

numBytes The number of bytes in the buffer.

6.777.3.14 `void activemq::wireformat::stomp::StompFrame::setCommand (const std::string & cmd) [inline]`

Sets the command for this stomp frame.

Parameters

cmd command The command to be set.

6.777.3.15 `void activemq::wireformat::stomp::StompFrame::setProperty (const std::string & name, const std::string & value) [inline]`

Sets the property given to the value specified in this Frame's Properties.

Parameters

name - Name of the property.

value - Value to set the property to.

6.777.3.16 `void activemq::wireformat::stomp::StompFrame::toStream (decaf::io::DataOutputStream * stream) const throw (decaf::io::IOException)`

Writes this Frame to an OutputStream in the Stomp Wire Format.

Parameters

stream - The stream to write the Frame to.

Exceptions

IOException if an error occurs while reading the Frame.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompFrame.h`

6.778 activemq::wireformat::stomp::StompHelper Class Reference

Utility Methods used when marshaling to and from StompFrame's.

`#include <src/main/activemq/wireformat/stomp/StompHelper.h>`

Public Member Functions

- **StompHelper** ()
- virtual **~StompHelper** ()
- void **convertProperties** (const **Pointer**< **StompFrame** > &frame, const **Pointer**< **Message** > &message)

Converts the Headers in a Stomp Frame into Headers in the given Message Command.
- void **convertProperties** (const **Pointer**< **Message** > &message, const **Pointer**< **StompFrame** > &frame)

*Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 3398).*
- **Pointer**< **ActiveMQDestination** > **convertDestination** (const std::string &destination)

Converts from a Stomp Destination to an ActiveMQDestination.
- std::string **convertDestination** (const **Pointer**< **ActiveMQDestination** > &destination)

Converts from a ActiveMQDestination to a Stomp Destination Name.
- std::string **convertMessageId** (const **Pointer**< **MessageId** > &messageId)

Converts a MessageId instance to a Stomp MessageId String.
- **Pointer**< **MessageId** > **convertMessageId** (const std::string &messageId)

Converts a Stomp MessageId string to a MessageId.
- std::string **convertConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)

Converts a ConsumerId instance to a Stomp ConsumerId String.
- **Pointer**< **ConsumerId** > **convertConsumerId** (const std::string &consumerId)

Converts a Stomp ConsumerId string to a ConsumerId.
- std::string **convertProducerId** (const **Pointer**< **ProducerId** > &producerId)

Converts a ProducerId instance to a Stomp ProducerId String.
- **Pointer**< **ProducerId** > **convertProducerId** (const std::string &producerId)

Converts a Stomp ProducerId string to a ProducerId.
- std::string **convertTransactionId** (const **Pointer**< **TransactionId** > &transactionId)

Converts a TransactionId instance to a Stomp TransactionId String.
- **Pointer**< **TransactionId** > **convertTransactionId** (const std::string &transactionId)

Converts a Stomp TransactionId string to a TransactionId.

6.778.1 Detailed Description

Utility Methods used when marshaling to and from StompFrame's.

Since

3.0

6.778.2 Constructor & Destructor Documentation

6.778.2.1 `activemq::wireformat::stomp::StompHelper::StompHelper () [inline]`

6.778.2.2 `virtual activemq::wireformat::stomp::StompHelper::~~StompHelper () [inline, virtual]`

6.778.3 Member Function Documentation

6.778.3.1 `std::string activemq::wireformat::stomp::StompHelper::convertConsumerId (const Pointer< ConsumerId > & consumerId)`

Converts a ConsumerId instance to a Stomp ConsumerId String.

Parameters

consumerId - the Consumer instance to convert.

Returns

a Stomp Consumer Id String.

6.778.3.2 `Pointer<ConsumerId> activemq::wireformat::stomp::StompHelper::convertConsumerId (const std::string & consumerId)`

Converts a Stomp ConsumerId string to a ConsumerId.

Parameters

consumerId - the String Consumer Id to convert.

Returns

Pointer to a new ConsumerId.

6.778.3.3 `std::string activemq::wireformat::stomp::StompHelper::convertDestination (const Pointer< ActiveMQDestination > & destination)`

Converts from a ActiveMQDestination to a Stomp Destination Name.

Parameters

destination - The ActiveMQDestination to Convert

Returns

the Stomp String name that defines the destination.

6.778.3.4 `Pointer<ActiveMQDestination> activemq::wireformat::stomp::StompHelper::convertDestination (const std::string & destination)`

Converts from a Stomp Destination to an ActiveMQDestination.

Parameters

destination - The Stomp Destination name string.

Returns

Pointer to a new ActiveMQDestination.

6.778.3.5 `std::string activemq::wireformat::stomp::StompHelper::convertMessageId (const Pointer< MessageId > & messageId)`

Converts a MessageId instance to a Stomp MessageId String.

Parameters

messageId - the MessageId instance to convert.

Returns

a Stomp Message Id String.

6.778.3.6 `Pointer<MessageId> activemq::wireformat::stomp::StompHelper::convertMessageId (const std::string & messageId)`

Converts a Stomp MessageId string to a MessageId.

Parameters

messageId - the String message Id to convert.

Returns

Pointer to a new MessageId.

6.778.3.7 `std::string activemq::wireformat::stomp::StompHelper::convertProducerId (const Pointer< ProducerId > & producerId)`

Converts a ProducerId instance to a Stomp ProducerId String.

Parameters

producerId - the Producer instance to convert.

Returns

a Stomp Producer Id String.

6.778.3.8 `Pointer<ProducerId> activemq::wireformat::stomp::StompHelper::convertProducerId (const std::string & producerId)`

Converts a Stomp ProducerId string to a ProducerId.

Parameters

producerId - the String Producer Id to convert.

Returns

Pointer to a new ProducerId.

6.778.3.9 `void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< Message > & message, const Pointer< StompFrame > & frame)`

Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 3398).

Parameters

message - The message to move the Headers to.

frame - The frame to extract headers from.

6.778.3.10 `void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< StompFrame > & frame, const Pointer< Message > & message)`

Converts the Headers in a Stomp Frame into Headers in the given Message Command.

Parameters

frame - The frame to extract headers from.

message - The message to move the Headers to.

6.778.3.11 `Pointer<TransactionId> activemq::wireformat::stomp::StompHelper::convertTransactionId (const std::string & transactionId)`

Converts a Stomp TransactionId string to a TransactionId.

Parameters

transactionId - the String Transaction Id to convert.

Returns

Pointer to a new TransactionId.

6.778.3.12 std::string activemq::wireformat::stomp::StompHelper::convertTransactionId (const Pointer< TransactionId > & transactionId)

Converts a TransactionId instance to a Stomp TransactionId String.

Parameters

transactionId - the Transaction instance to convert.

Returns

a Stomp Transaction Id String.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/StompHelper.h

6.779 activemq::wireformat::stomp::StompWireFormat Class Reference

```
#include <src/main/activemq/wireformat/stomp/StompWireFormat.h>
```

Inheritance diagram for activemq::wireformat::stomp::StompWireFormat:

Public Member Functions

- **StompWireFormat** ()
- virtual **~StompWireFormat** ()
- virtual void **marshal** (const **Pointer< commands::Command >** &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out) throw (**decaf::io::IOException**)
Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.
- virtual **Pointer< commands::Command >** **unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in) throw (**decaf::io::IOException**)
Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.
- virtual void **setVersion** (int version AMQCPP_UNUSED)
Set the Version.
- virtual int **getVersion** () const
Get the Version.
- virtual bool **inReceive** () const
Is there a Message being unmarshaled?

- virtual bool **hasNegotiator** () const

*Returns true if this **WireFormat** (p. 3712) has a Negotiator that needs to wrap the Transport that uses it.*

- virtual **Pointer< transport::Transport > createNegotiator**
(const **Pointer< transport::Transport > &transport**) throw (decaf::lang::exceptions::UnsupportedOperationException)

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

6.779.1 Constructor & Destructor Documentation

6.779.1.1 **activemq::wireformat::stomp::StompWireFormat::StompWireFormat (**
)

6.779.1.2 **virtual**
activemq::wireformat::stomp::StompWireFormat::~~StompWireFormat (
) [virtual]

6.779.2 Member Function Documentation

6.779.2.1 **virtual Pointer<transport::Transport> ac-**
tivemq::wireformat::stomp::StompWireFormat::createNegotiator
(const Pointer< transport::Transport > & transport) throw (
decaf::lang::exceptions::UnsupportedOperationException) [virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

Returns

new instance of a **WireFormatNegotiator** (p. 3751).

Exceptions

***UnsupportedOperationException** if the **WireFormat** (p. 3712) doesn't have a Negotiator.*

Implements **activemq::wireformat::WireFormat** (p. 3714).

6.779.2.2 **virtual int activemq::wireformat::stomp::StompWireFormat::getVersion (**
) const [inline, virtual]

Get the Version.

Returns

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 3714).

6.779.2.3 `virtual bool activemq::wireformat::stomp::StompWireFormat::hasNegotiator () const` [inline, virtual]

Returns true if this **WireFormat** (p. 3712) has a Negotiator that needs to wrap the Transport that uses it.

Returns

true if the **WireFormat** (p. 3712) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3714).

6.779.2.4 `virtual bool activemq::wireformat::stomp::StompWireFormat::inReceive () const` [inline, virtual]

Is there a Message being unmarshaled?

Returns

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 3715).

6.779.2.5 `virtual void activemq::wireformat::stomp::StompWireFormat::marshal (const Pointer< commands::Command > & command, const activemq::transport::Transport * transport, decaf::io::DataOutputStream * out) throw (decaf::io::IOException)` [virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters

command The Command to Marshal to the output stream.

transport The Transport that initiated this marshal call.

out The output stream to write the command to.

Exceptions

IOException

Implements **activemq::wireformat::WireFormat** (p. 3715).

6.779.2.6 `virtual void activemq::wireformat::stomp::StompWireFormat::setVersion (int version AMQCPP_UNUSED)` [inline, virtual]

Set the Version.

Parameters

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 3715).

```

6.779.2.7 virtual Pointer<commands::Command> ac-
    tivemq::wireformat::stomp::StompWireFormat::unmarshal
    ( const activemq::transport::Transport * transport,
      decaf::io::DataInputStream * in ) throw ( decaf::io::IOException )
    [virtual]

```

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

Returns a Pointer to the newly unmarshaled Command.

Parameters

- transport* - Pointer to the transport that is making this request.
- in* - the input stream to read the command from.

Returns

the newly marshaled Command, caller owns the pointer

Exceptions

IOException

Implements **activemq::wireformat::WireFormat** (p. 3716).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompWireFormat.h**

6.780 activemq::wireformat::stomp::StompWireFormatFactory Class Reference

Factory used to create the Stomp Wire Format instance.

```
#include <src/main/activemq/wireformat/stomp/StompWireFormatFactory.h>
```

Inheritance diagram for **activemq::wireformat::stomp::StompWireFormatFactory**:

Public Member Functions

- **StompWireFormatFactory** ()
- virtual **~StompWireFormatFactory** ()
- virtual **Pointer< WireFormat > createWireFormat** (const **decaf::util::Properties** &properties) throw (**decaf::lang::exceptions::IllegalStateException**)

*Creates a new **WireFormat** (p. 3712) Object passing it a set of properties from which it can obtain any optional settings.*

6.780.1 Detailed Description

Factory used to create the Stomp Wire Format instance.

6.780.2 Constructor & Destructor Documentation

- 6.780.2.1** `activemq::wireformat::stomp::StompWireFormatFactory::StompWireFormatFactory () [inline]`
- 6.780.2.2** `virtual
activemq::wireformat::stomp::StompWireFormatFactory::~~StompWireFormatFactory () [inline, virtual]`

6.780.3 Member Function Documentation

- 6.780.3.1** `virtual Pointer<WireFormat> activemq::wireformat::stomp::StompWireFormatFactory::createWireFormat (const decaf::util::Properties & properties) throw (decaf::lang::exceptions::IllegalStateException) [virtual]`

Creates a new **WireFormat** (p. 3712) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

properties - the Properties for this **WireFormat** (p. 3712)

Implements **activemq::wireformat::WireFormatFactory** (p. 3717).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompWireFormatFactory.h`

6.781 cms::Stoppable Class Reference

Interface for a class that implements the stop method.

```
#include <src/main/cms/Stoppable.h>
```

Inheritance diagram for cms::Stoppable:

Public Member Functions

- `virtual ~Stoppable ()`
- `virtual void stop ()=0 throw (CMSException)`
Stops this service.

6.781.1 Detailed Description

Interface for a class that implements the stop method. An object that implements this interface implies that it will halt all operations that result in events being propagated to external users, internally the Object can continue to process data but not events will be generated to clients and methods that return data will not return valid results until the object is started again.

Since

1.0

6.781.2 Constructor & Destructor Documentation

6.781.2.1 virtual `cms::Stoppable::~~Stoppable ()` [inline, virtual]

6.781.3 Member Function Documentation

6.781.3.1 virtual void `cms::Stoppable::stop ()` throw (`CMSEException`) [pure virtual]

Stops this service.

Exceptions

CMSEException (p. 1074) - if an internal error occurs while stopping the Service.

The documentation for this class was generated from the following file:

- `src/main/cms/Stoppable.h`

6.782 decaf::util::logging::StreamHandler Class Reference

Stream based logging **Handler** (p. 1852).

```
#include <src/main/decaf/util/logging/StreamHandler.h>
```

Inheritance diagram for `decaf::util::logging::StreamHandler`:

Public Member Functions

- **StreamHandler** ()
*Create a **StreamHandler** (p. 3412), with no current output stream.*
- **StreamHandler** (`decaf::io::OutputStream` *stream, `Formatter` *formatter)
*Create a **StreamHandler** (p. 3412), with no current output stream.*
- virtual `~StreamHandler` ()
- virtual void **close** () throw (`decaf::io::IOException`)
Close the current output stream.
- virtual void **flush** ()
Flush the Handler's output, clears any buffers.
- virtual void **publish** (const `LogRecord` &record)
*Publish the Log Record to this **Handler** (p. 1852).*

- virtual bool **isLoggable** (const **LogRecord** &record) const

*Check if this **Handler** (p. 1852) would actually log a given **LogRecord** (p. 2261).*

Protected Member Functions

- virtual void **setOutputStream** (decaf::io::OutputStream *stream) throw (decaf::lang::exceptions::NullPointerException)

Change the output stream.

- void **close** (bool closeStream)

Closes this handler, but the underlying output stream is only closed if closeStream is true.

6.782.1 Detailed Description

Stream based logging **Handler** (p.1852). This is primarily intended as a base class or support class to be used in implementing other logging Handlers.

LogRecords are published to a given decaf::io::OutputStream (p. 2718).

Configuration: By default each **StreamHandler** (p.3412) is initialized using the following **Log-Manager** (p. 2254) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

* decaf.util.logging.StreamHandler.level specifies the default level for the **Handler** (p.1852) (defaults to **Level.INFO** (p.2190)). * decaf.util.logging.StreamHandler.filter specifies the name of a **Filter** (p.1770) class to use (defaults to no **Filter** (p.1770)). * decaf.util.logging.StreamHandler.formatter specifies the name of a **Formatter** (p.1838) class to use (defaults to **decaf.util.logging.SimpleFormatter** (p. 3278)).

Since

1.0

6.782.2 Constructor & Destructor Documentation

6.782.2.1 decaf::util::logging::StreamHandler::StreamHandler ()

Create a **StreamHandler** (p. 3412), with no current output stream.

6.782.2.2 decaf::util::logging::StreamHandler::StreamHandler (decaf::io::OutputStream * stream, Formatter * formatter)

Create a **StreamHandler** (p. 3412), with no current output stream.

6.782.2.3 `virtual decaf::util::logging::StreamHandler::~~StreamHandler ()`
[virtual]

6.782.3 Member Function Documentation

6.782.3.1 `virtual void decaf::util::logging::StreamHandler::close () throw (decaf::io::IOException)` [virtual]

Close the current output stream.

The close method will perform a flush and then close the **Handler** (p.1852). After close has been called this **Handler** (p.1852) should no longer be used. Method calls may either be silently ignored or may throw runtime exceptions.

Exceptions

IOException if an I/O error occurs.

Implements `decaf::io::Closeable` (p.1066).

Reimplemented in `decaf::util::logging::ConsoleHandler` (p.1301).

6.782.3.2 `void decaf::util::logging::StreamHandler::close (bool closeStream)`
[protected]

Closes this handler, but the underlying output stream is only closed if `closeStream` is true.

Parameters

closeStream whether to close the underlying output stream.

6.782.3.3 `virtual void decaf::util::logging::StreamHandler::flush ()` [virtual]

Flush the Handler's output, clears any buffers.

Implements `decaf::util::logging::Handler` (p.1853).

6.782.3.4 `virtual bool decaf::util::logging::StreamHandler::isLoggable (const LogRecord & record) const` [virtual]

Check if this **Handler** (p.1852) would actually log a given **LogRecord** (p.2261).

Parameters

record LogRecord (p.2261) to check

Returns

true if the record can be logged with current settings.

Reimplemented from `decaf::util::logging::Handler` (p.1854).

6.782.3.5 virtual void decaf::util::logging::StreamHandler::publish (const LogRecord & *record*) [virtual]

Publish the Log Record to this **Handler** (p. 1852).

Parameters

record The LogRecord (p. 2261) to Publish

Implements **decaf::util::logging::Handler** (p. 1854).

Reimplemented in **decaf::util::logging::ConsoleHandler** (p. 1302).

6.782.3.6 virtual void decaf::util::logging::StreamHandler::setOutputStream (decaf::io::OutputStream * *stream*) throw (decaf::lang::exceptions::NullPointerException) [protected, virtual]

Change the output stream.

If there is a current output stream then the Formatter's tail string is written and the stream is flushed and closed. Then the output stream is replaced with the new output stream.

Parameters

stream The new output stream. May not be NULL.

Exceptions

NullPointerException if the passed stream is NULL.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**StreamHandler.h**

6.783 cms::StreamMessage Class Reference

Interface for a **StreamMessage** (p. 3415).

```
#include <src/main/cms/StreamMessage.h>
```

Inheritance diagram for cms::StreamMessage:

Public Member Functions

- virtual **~StreamMessage** ()
- virtual bool **readBoolean** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)
Reads a Boolean from the Stream message stream.
- virtual void **writeBoolean** (bool value)=0 throw (cms::MessageNotWriteableException, cms::CMSException)

Writes a boolean to the Stream message stream as a 1-byte value.

- virtual unsigned char **readByte** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a Byte from the Stream message stream.

- virtual void **writeByte** (unsigned char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a byte to the Stream message stream as a 1-byte value.

- virtual int **readBytes** (std::vector< unsigned char > &value) const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a byte array from the Stream message stream.

- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

- virtual int **readBytes** (unsigned char *buffer, int length) const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a portion of the Stream message stream.

- virtual void **writeBytes** (const unsigned char *value, int offset, int length)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a portion of a byte array to the Stream message stream.

- virtual char **readChar** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a Char from the Stream message stream.

- virtual void **writeChar** (char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a char to the Stream message stream as a 1-byte value.

- virtual float **readFloat** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit float from the Stream message stream.

- virtual void **writeFloat** (float value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a float to the Stream message stream as a 4 byte value.

- virtual double **readDouble** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit double from the Stream message stream.

- virtual void **writeDouble** (double value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a double to the Stream message stream as a 8 byte value.

- virtual short **readShort** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit signed short from the Stream message stream.

- virtual void **writeShort** (short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed short to the Stream message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit unsigned short from the Stream message stream.

- virtual void **writeUnsignedShort** (unsigned short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a unsigned short to the Stream message stream as a 2 byte value.

- virtual int **readInt** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit signed integer from the Stream message stream.

- virtual void **writeInt** (int value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed int to the Stream message stream as a 4 byte value.

- virtual long long **readLong** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit long from the Stream message stream.

- virtual void **writeLong** (long long value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a long long to the Stream message stream as a 8 byte value.

- virtual std::string **readString** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads an ASCII String from the Stream message stream.

- virtual void **writeString** (const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes an ASCII String to the Stream message stream.

6.783.1 Detailed Description

Interface for a **StreamMessage** (p. 3415). The stream Messages provides a **Message** (p. 2375) type whose body is a stream of self describing primitive types. The primitive values are read and written using accessors specific to the given types.

StreamMessage (p. 3415) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSException** (p. 1074). The string-to-primitive conversions may throw a runtime exception if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X									X
byte		X	X		X	X				X
short			X		X	X				X
char				X						X
int					X	X				X
long						X				X
float							X	X		X
double								X	X	X
String	X	X	X		X	X	X	X	X	
byte[]										X

Since

1.3

6.783.2 Constructor & Destructor Documentation

6.783.2.1 `virtual cms::StreamMessage::~StreamMessage () [inline, virtual]`

6.783.3 Member Function Documentation

6.783.3.1 `virtual bool cms::StreamMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a Boolean from the Stream message stream.

Returns

boolean value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.2 virtual unsigned char cms::StreamMessage::readByte () const
throw (cms::MessageEOFException, cms::MessageFormatException,
cms::MessageNotReadableException, cms::CMSException) [pure
virtual]

Reads a Byte from the Stream message stream.

Returns

unsigned char value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.3 virtual int cms::StreamMessage::readBytes (std::vector< unsigned
char > & value) const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [pure virtual]

Reads a byte array from the Stream message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

value buffer to place data in

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.4 `virtual int cms::StreamMessage::readBytes (unsigned char *
buffer, int length) const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [pure virtual]`

Reads a portion of the Stream message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an **CMSException** (p.1074) is thrown. No bytes will be read from the stream for this exception case.

Parameters

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.5 `virtual char cms::StreamMessage::readChar () const throw
(cms::MessageEOFException, cms::MessageFormatException,
cms::MessageNotReadableException, cms::CMSException) [pure
virtual]`

Reads a Char from the Stream message stream.

Returns

char value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.6 virtual double cms::StreamMessage::readDouble () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a 64 bit double from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.7 virtual float cms::StreamMessage::readFloat () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a 32 bit float from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.8 virtual int cms::StreamMessage::readInt () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a 32 bit signed integer from the Stream message stream.

Returns

int value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.9 `virtual long long cms::StreamMessage::readLong () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 64 bit long from the Stream message stream.

Returns

long long value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.10 `virtual short cms::StreamMessage::readShort () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 16 bit signed short from the Stream message stream.

Returns

short value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.11 `virtual std::string cms::StreamMessage::readString () const
 throw (cms::MessageEOFException, cms::MessageFormatException,
 cms::MessageNotReadableException, cms::CMSException) [pure
 virtual]`

Reads an ASCII String from the Stream message stream.

Returns

String from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.12 `virtual unsigned short cms::StreamMessage::readUnsignedShort
 () const throw (cms::MessageEOFException,
 cms::MessageFormatException, cms::MessageNotReadableException,
 cms::CMSException) [pure virtual]`

Reads a 16 bit unsigned short from the Stream message stream.

Returns

unsigned short value from stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2492) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2493) - if this type conversion is invalid.

MessageNotReadableException (p. 2548) - if the message is in write-only mode.

6.783.3.13 `virtual void cms::StreamMessage::writeBoolean (bool value) throw
 (cms::MessageNotWriteableException, cms::CMSException) [pure
 virtual]`

Writes a boolean to the Stream message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

value boolean to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.783.3.14 `virtual void cms::StreamMessage::writeByte (unsigned char value)
throw (cms::MessageNotWriteableException, cms::CMSException)
[pure virtual]`

Writes a byte to the Stream message stream as a 1-byte value.

Parameters

value byte to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.783.3.15 `virtual void cms::StreamMessage::writeBytes (const
unsigned char * value, int offset, int length) throw (cms::MessageNotWriteableException, cms::CMSException) [pure
virtual]`

Writes a portion of a byte array to the Stream message stream.

size as the number of bytes to write.

Parameters

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.783.3.16 `virtual void cms::StreamMessage::writeBytes (const
std::vector< unsigned char > & value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure
virtual]`

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

Parameters

value bytes to write to the stream

Exceptions

CMSEException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.783.3.17 `virtual void cms::StreamMessage::writeChar (char value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Writes a char to the Stream message stream as a 1-byte value.

Parameters

value char to write to the stream

Exceptions

CMSEException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.783.3.18 `virtual void cms::StreamMessage::writeDouble (double value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Writes a double to the Stream message stream as a 8 byte value.

Parameters

value double to write to the stream

Exceptions

CMSEException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.783.3.19 `virtual void cms::StreamMessage::writeFloat (float value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Writes a float to the Stream message stream as a 4 byte value.

Parameters

value float to write to the stream

Exceptions

CMSEException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.783.3.20 `virtual void cms::StreamMessage::writeInt (int value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Writes a signed int to the Stream message stream as a 4 byte value.

Parameters

value signed int to write to the stream

Exceptions

CMSEException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.783.3.21 `virtual void cms::StreamMessage::writeLong (long long value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Writes a long long to the Stream message stream as a 8 byte value.

Parameters

value signed long long to write to the stream

Exceptions

CMSEException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.783.3.22 `virtual void cms::StreamMessage::writeShort (short value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Writes a signed short to the Stream message stream as a 2 byte value.

Parameters

value signed short to write to the stream

Exceptions

CMSEException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.783.3.23 `virtual void cms::StreamMessage::writeString (const std::string
& value) throw (cms::MessageNotWriteableException,
cms::CMSException) [pure virtual]`

Writes an ASCII String to the Stream message stream.

Parameters

value String to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

6.783.3.24 `virtual void cms::StreamMessage::writeUnsignedShort (unsigned
short value) throw (cms::MessageNotWriteableException,
cms::CMSException) [pure virtual]`

Writes a unsigned short to the Stream message stream as a 2 byte value.

Parameters

value unsigned short to write to the stream

Exceptions

CMSException (p. 1074) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode.

The documentation for this class was generated from the following file:

- `src/main/cms/StreamMessage.h`

6.784 decaf::lang::String Class Reference

The **String** (p. 3427) class represents an immutable sequence of chars.

`#include <src/main/decaf/lang/String.h>`

Inheritance diagram for decaf::lang::String:

Public Member Functions

- **String** ()
*Creates a new empty **String** (p. 3427) object.*

- **String** (const std::string &source)

Create a new **String** (p. 3427) object that represents the given STL string.

- virtual ~**String** ()
- bool **isEmpty** () const
- virtual int **length** () const

Returns

the length of the underlying character sequence.

- virtual char **charAt** (int index) const throw (lang::exceptions::IndexOutOfBoundsException)

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

Parameters

index - position to return the char at.

Returns

the char at the given position

Exceptions

IndexOutOfBoundsException if index is > than **length()** (p. 1054) or negative

- virtual **CharSequence** * **subSequence** (int start, int end) const throw (lang::exceptions::IndexOutOfBoundsException)

Returns a new **CharSequence** (p. 1053) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.

Parameters

start - the start index, inclusive
end - the end index, exclusive

Returns

a new **CharSequence** (p. 1053)

Exceptions

IndexOutOfBoundsException if start or end > **length()** (p. 1054) or start or end are negative.

- virtual std::string **toString** () const

Returns

the string representation of this **CharSequence** (p. 1053)

6.784.1 Detailed Description

The **String** (p. 3427) class represents an immutable sequence of chars.

Since

1.0

6.784.2 Constructor & Destructor Documentation

6.784.2.1 decaf::lang::String::String ()

Creates a new empty **String** (p. 3427) object.

6.784.2.2 decaf::lang::String::String (const std::string & *source*)

Create a new **String** (p. 3427) object that represents the given STL string.

Parameters

source The string to copy into this new **String** (p. 3427) object.

6.784.2.3 virtual decaf::lang::String::~~String () [virtual]

6.784.3 Member Function Documentation

6.784.3.1 virtual char decaf::lang::String::charAt (int *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

Parameters

index - position to return the char at.

Returns

the char at the given position

Exceptions

IndexOutOfBoundsException if index is > than **length()** (p. 1054) or negative

Implements **decaf::lang::CharSequence** (p. 1054).

6.784.3.2 bool decaf::lang::String::isEmpty () const

Returns

true if the length of this **String** (p. 3427) is zero.

6.784.3.3 virtual int decaf::lang::String::length () const [virtual]

Returns

the length of the underlying character sequence.

Implements **decaf::lang::CharSequence** (p. 1054).

6.784.3.4 `virtual CharSequence* decaf::lang::String::subSequence (int start, int end) const throw (lang::exceptions::IndexOutOfBoundsException)` [virtual]

Returns a new **CharSequence** (p. 1053) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index `end - 1`. The length (in chars) of the returned sequence is `end - start`, so if `start == end` then an empty sequence is returned.

Parameters

start - the start index, inclusive

end - the end index, exclusive

Returns

a new **CharSequence** (p. 1053)

Exceptions

IndexOutOfBoundsException if `start` or `end > length()` (p. 1054) or `start` or `end` are negative.

Implements **decaf::lang::CharSequence** (p. 1054).

6.784.3.5 `virtual std::string decaf::lang::String::toString () const` [virtual]

Returns

the string representation of this **CharSequence** (p. 1053)

Implements **decaf::lang::CharSequence** (p. 1055).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/String.h`

6.785 decaf::util::StringTokenizer Class Reference

```
#include <src/main/decaf/util/StringTokenizer.h>
```

Public Member Functions

- **StringTokenizer** (const std::string &str, const std::string &delim=" \t\n\r\f", bool returnDelims=false)

Constructs a string tokenizer for the specified string.

- virtual **~StringTokenizer** ()
- virtual int **countTokens** () const

Calculates the number of times that this tokenizer's `nextToken` method can be called before it generates an exception.

- virtual bool **hasMoreTokens** () const
Tests if there are more tokens available from this tokenizer's string.
- virtual std::string **nextToken** () throw (lang::exceptions::NoSuchElementException)
Returns the next token from this string tokenizer.
- virtual std::string **nextToken** (const std::string &delim) throw (lang::exceptions::NoSuchElementException)
Returns the next token in this string tokenizer's string.
- virtual unsigned int **toArray** (std::vector< std::string > &array)
Grab all remaining tokens in the String and return them in the vector that is passed in by reference.
- virtual void **reset** (const std::string &str="", const std::string &delim="", bool returnDelims=false)
Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.

6.785.1 Constructor & Destructor Documentation

6.785.1.1 decaf::util::StringTokenizer::StringTokenizer (const std::string & *str*, const std::string & *delim* = " \t\n\r\f", bool *returnDelims* = false)

Constructs a string tokenizer for the specified string.

All characters in the delim argument are the delimiters for separating tokens.

If the returnDelims flag is true, then the delimiter characters are also returned as tokens. Each delimiter is returned as a string of length one. If the flag is false, the delimiter characters are skipped and only serve as separators between tokens.

Note that if delim is "", this constructor does not throw an exception. However, trying to invoke other methods on the resulting **StringTokenizer** (p. 3430) may result in an Exception.

Parameters

str - The string to tokenize

delim - String containing the delimiters

returnDelims - boolean indicating if the delimiters are returned as tokens

6.785.1.2 virtual decaf::util::StringTokenizer::~~StringTokenizer () [virtual]

6.785.2 Member Function Documentation

6.785.2.1 virtual int decaf::util::StringTokenizer::countTokens () const [virtual]

Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.

The current position is not advanced.

Returns

Count of remaining tokens

6.785.2.2 `virtual bool decaf::util::StringTokenizer::hasMoreTokens () const`
[virtual]

Tests if there are more tokens available from this tokenizer's string.

Returns

true if there are more tokens remaining

6.785.2.3 `virtual std::string decaf::util::StringTokenizer::nextToken (const`
`std::string & delim) throw (lang::exceptions::NoSuchElementException`
`) [virtual]`

Returns the next token in this string tokenizer's string.

First, the set of characters considered to be delimiters by this **StringTokenizer** (p. 3430) object is changed to be the characters in the string *delim*. Then the next token in the string after the current position is returned. The current position is advanced beyond the recognized token. The new delimiter set remains the default after this call.

Parameters

delim - string containing the new set of delimiters

Returns

next string in the token list

Exceptions

NoSuchElementException

6.785.2.4 `virtual std::string decaf::util::StringTokenizer::nextToken () throw (`
`lang::exceptions::NoSuchElementException) [virtual]`

Returns the next token from this string tokenizer.

Returns

string value of next token

Exceptions

NoSuchElementException

6.785.2.5 `virtual void decaf::util::StringTokenizer::reset (const std::string & str = "", const std::string & delim = "", bool returnDelims = false) [virtual]`

Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.

This allows this object to be reused, the caller need not create a new instance every time a String needs tokenizing. If set the string param will reset the string that this Tokenizer is working on. If set to "" no change is made. If set the delim param will reset the string that this Tokenizer is using to tokenize the string. If set to "", no change is made. If set the return Delims will set if this Tokenizer will return delimiters as tokens. Defaults to false.

Parameters

str - New String to tokenize or "", defaults to ""

delim - New Delimiter String to use or "", defaults to ""

returnDelims - Should the Tokenizer return delimiters as Tokens, default false

6.785.2.6 `virtual unsigned int decaf::util::StringTokenizer::toArray (std::vector< std::string > & array) [virtual]`

Grab all remaining tokens in the String and return them in the vector that is passed in by reference.

Parameters

array - vector to place token strings in

Returns

number of string placed into the vector

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StringTokenizer.h`

6.786 activemq::commands::SubscriptionInfo Class Reference

```
#include <src/main/activemq/commands/SubscriptionInfo.h>
```

Inheritance diagram for activemq::commands::SubscriptionInfo:

Public Member Functions

- `SubscriptionInfo ()`
- `virtual ~SubscriptionInfo ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.

- virtual **SubscriptionInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &selector)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** ()
- virtual void **setSubscribedDestination** (const **Pointer**< **ActiveMQDestination** > &subscribedDestination)

Static Public Attributes

- static const unsigned char **ID_SUBSCRIPTIONINFO** = 55

Protected Attributes

- std::string **clientId**
- **Pointer**< **ActiveMQDestination** > **destination**
- std::string **selector**
- std::string **subscriptionName**
- **Pointer**< **ActiveMQDestination** > **subscribedDestination**

6.786.1 Constructor & Destructor Documentation

6.786.1.1 `activemq::commands::SubscriptionInfo::SubscriptionInfo ()`

6.786.1.2 `virtual activemq::commands::SubscriptionInfo::~~SubscriptionInfo ()`
[virtual]

6.786.2 Member Function Documentation

6.786.2.1 `virtual SubscriptionInfo* activemq::commands::SubscriptionInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.786.2.2 `virtual void activemq::commands::SubscriptionInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1555).

6.786.2.3 `virtual bool activemq::commands::SubscriptionInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1556).

6.786.2.4 `virtual const std::string& activemq::commands::SubscriptionInfo::getClientId () const [virtual]`

6.786.2.5 `virtual std::string& activemq::commands::SubscriptionInfo::getClientId () [virtual]`

6.786.2.6 `virtual unsigned char activemq::commands::SubscriptionInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

Implements **activemq::commands::DataStructure** (p. 1557).

- 6.786.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination () const` [virtual]
- 6.786.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination ()` [virtual]
- 6.786.2.9 `virtual const std::string& activemq::commands::SubscriptionInfo::getSelector () const` [virtual]
- 6.786.2.10 `virtual std::string& activemq::commands::SubscriptionInfo::getSelector ()` [virtual]
- 6.786.2.11 `virtual std::string& activemq::commands::SubscriptionInfo::getSubscriptionName ()` [virtual]
- 6.786.2.12 `virtual const std::string& activemq::commands::SubscriptionInfo::getSubscriptionName () const` [virtual]
- 6.786.2.13 `virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination () const` [virtual]
- 6.786.2.14 `virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination ()` [virtual]
- 6.786.2.15 `virtual void activemq::commands::SubscriptionInfo::setClientId (const std::string & clientId)` [virtual]
- 6.786.2.16 `virtual void activemq::commands::SubscriptionInfo::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.786.2.17 `virtual void activemq::commands::SubscriptionInfo::setSelector (const std::string & selector)` [virtual]
- 6.786.2.18 `virtual void activemq::commands::SubscriptionInfo::setSubscriptionName (const std::string & subscriptionName)` [virtual]
- 6.786.2.19 `virtual void activemq::commands::SubscriptionInfo::setSubscribedDestination (const Pointer< ActiveMQDestination > & subscribedDestination)` [virtual]
- 6.786.2.20 `virtual std::string activemq::commands::SubscriptionInfo::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 767).

6.786.3 Field Documentation

6.786.3.1 `std::string activemq::commands::SubscriptionInfo::clientId` [protected]

6.786.3.2 `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::destination` [protected]

6.786.3.3 `const unsigned char activemq::commands::SubscriptionInfo::ID_ - SUBSCRIPTIONINFO = 55` [static]

6.786.3.4 `std::string activemq::commands::SubscriptionInfo::selector` [protected]

6.786.3.5 `std::string activemq::commands::SubscriptionInfo::subscriptionName` [protected]

6.786.3.6 `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::subscribedDestination` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SubscriptionInfo.h`

6.787 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for `SubscriptionInfoMarshaller` (p. 3438).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller`:

Public Member Functions

- `SubscriptionInfoMarshaller ()`
- `virtual ~SubscriptionInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.787.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3438). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.787.2 Constructor & Destructor Documentation

6.787.2.1 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller () [inline]

6.787.2.2 virtual
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller () [inline, virtual]

6.787.3 Member Function Documentation

6.787.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.787.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.787.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.787.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

```
6.787.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

```
6.787.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```

6.787.3.7 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h`

6.788 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for `SubscriptionInfoMarshaller` (p. 3442).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller`:

Public Member Functions

- `SubscriptionInfoMarshaller ()`
- `virtual ~SubscriptionInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.788.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3442). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.788.2 Constructor & Destructor Documentation

6.788.2.1 **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller** () [inline]

6.788.2.2 **virtual**
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller () [inline, virtual]

6.788.3 Member Function Documentation

6.788.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.788.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::getDataStructureType() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.788.3.3 virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1518).

6.788.3.4 virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseUnmarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.788.3.5 virtual int ac-
tivismq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal1
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.788.3.6 virtual void ac-
tivismq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal2
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.788.3.7 virtual void ac-
tivismq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightUnmarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, decaf::io::DataInputStream * *dataIn*,
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h

6.789 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMar Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3446).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller**:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.789.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3446). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.789.2 Constructor & Destructor Documentation

6.789.2.1 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller () [inline]

6.789.2.2 virtual activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller () [inline, virtual]

6.789.3 Member Function Documentation

6.789.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.789.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.789.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.789.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.789.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.789.3.6 virtual void **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal2**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.789.3.7 virtual void **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h`

6.790 **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3450).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller**:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.790.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3450). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.790.2 Constructor & Destructor Documentation

6.790.2.1 `activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` [inline]

6.790.2.2 `virtual activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` [inline, virtual]

6.790.3 Member Function Documentation

6.790.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.790.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.790.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.790.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.790.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.790.3.6 virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.790.3.7 virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h

6.791 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3453).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller`:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.791.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3453). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.791.2 Constructor & Destructor Documentation

6.791.2.1 `activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` [inline]

6.791.2.2 `virtual activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` [inline, virtual]

6.791.3 Member Function Documentation

6.791.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.791.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.791.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.791.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.791.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.791.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.791.3.7 virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h

6.792 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3457).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.792.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3457). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.792.2 Constructor & Destructor Documentation

6.792.2.1 `activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` [inline]

6.792.2.2 `virtual activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` [inline, virtual]

6.792.3 Member Function Documentation

6.792.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.792.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.792.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.792.3.4 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.792.3.5 virtual int activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.792.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

6.792.3.7 virtual void **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h`

6.793 decaf::util::concurrent::Synchronizable Class Reference

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

```
#include <src/main/decaf/util/concurrent/Synchronizable.h>
```

Inheritance diagram for `decaf::util::concurrent::Synchronizable`:

Public Member Functions

- virtual `~Synchronizable ()`
- virtual void **lock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
*Attempts to **Lock** (p. 2228) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
*Waits on a signal from this object, which is generated by a call to *Notify*.*
- virtual void **wait** (long long millisecs)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
*Waits on a signal from this object, which is generated by a call to *Notify*.*
- virtual void **wait** (long long millisecs, int nanos)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
*Waits on a signal from this object, which is generated by a call to *Notify*.*
- virtual void **notify** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.793.1 Detailed Description

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Since

1.0

6.793.2 Constructor & Destructor Documentation

6.793.2.1 virtual decaf::util::concurrent::Synchronizable::~~Synchronizable ()
[inline, virtual]

6.793.3 Member Function Documentation

6.793.3.1 virtual void decaf::util::concurrent::Synchronizable::lock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2434), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3474), `decaf::io::InputStream` (p. 1913), `decaf::io::OutputStream` (p. 2721), `decaf::util::AbstractCollection< E >` (p. 150), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1149), `decaf::util::concurrent::Mutex` (p. 2605), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3377), `decaf::util::StlQueue< T >` (p. 3386), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 150), `decaf::util::AbstractCollection< Pointer< Synchroniza-tion > >` (p. 150), `decaf::util::AbstractCollection< Resource * >` (p. 150), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 150), `decaf::util::AbstractCollection< CompositeTask * >` (p. 150), `decaf::util::AbstractCollection< URI >` (p. 150), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 150), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 150), `decaf::util::AbstractCollection< Prim-itiveValueNode >` (p. 150), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 150), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 150), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 150), `decaf::util::AbstractCollection< cms::Destination * >` (p. 150), `decaf::util::AbstractCollection< cms::Session * >` (p. 150), `decaf::util::AbstractCollection< cms::Connection * >` (p. 150), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Mes-sageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1149), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1149), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1149), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1149), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1149), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1149), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3377), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3377), `decaf::util::StlMap< std::string, PrimitiveVal-ueNode >` (p. 3377), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3377), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com-mands::ProducerId::COMPARATOR >` (p. 3377), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3377), `decaf::util::StlMap< Pointer< commands::ConsumerId`

>, `ActiveMQConsumer *`, `commands::ConsumerId::COMPARATOR` > (p. 3377), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3377), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3377), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3377), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3377), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3377), `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3377), `decaf::util::StlQueue< Pointer< Transport > >` (p. 3386), `decaf::util::StlQueue< Pointer< MessageDispatch > >` (p. 3386), `decaf::util::StlQueue< Task >` (p. 3386), `decaf::util::StlQueue< Pointer< Command > >` (p. 3386), and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >` (p. 3386).

6.793.3.2 `virtual void decaf::util::concurrent::Synchronizable::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)` [pure virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2435), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3474), `decaf::io::InputStream` (p. 1914), `decaf::io::OutputStream` (p. 2722), `decaf::util::AbstractCollection< E >` (p. 150), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1149), `decaf::util::concurrent::Mutex` (p. 2606), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3377), `decaf::util::StlQueue< T >` (p. 3386), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 150), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 150), `decaf::util::AbstractCollection< Resource * >` (p. 150), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 150), `decaf::util::AbstractCollection< CompositeTask * >` (p. 150), `decaf::util::AbstractCollection< URI >` (p. 150), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 150), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 150), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 150), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 150), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 150), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 150), `decaf::util::AbstractCollection< cms::Destination * >` (p. 150), `decaf::util::AbstractCollection< cms::Session * >` (p. 150), `decaf::util::AbstractCollection< cms::Connection * >` (p. 150), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1149), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1149), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR`

> (p. 1149), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1149), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1149), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1149), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3377), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3377), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3377), decaf::util::StlMap< std::string, cms::Queue * > (p. 3377), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3377), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3377), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3377), decaf::util::StlMap< std::string, TransportFactory * > (p. 3377), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3377), decaf::util::StlMap< int, Pointer< Command > > (p. 3377), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3377), decaf::util::StlMap< std::string, CachedProducer * > (p. 3377), decaf::util::StlMap< std::string, cms::Topic * > (p. 3377), decaf::util::StlQueue< Pointer< Transport > > (p. 3386), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3386), decaf::util::StlQueue< Task > (p. 3386), decaf::util::StlQueue< Pointer< Command > > (p. 3386), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3386).

6.793.3.3 virtual void decaf::util::concurrent::Synchronizable::notifyAll
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2435), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 3474), **decaf::io::InputStream** (p. 1914), **decaf::io::OutputStream** (p. 2722), **decaf::util::AbstractCollection< E >** (p. 151), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1149), **decaf::util::concurrent::Mutex** (p. 2606), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3378), **decaf::util::StlQueue< T >** (p. 3387), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 151), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 151), **decaf::util::AbstractCollection< Resource * >** (p. 151), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 151), **decaf::util::AbstractCollection< CompositeTask * >** (p. 151), **decaf::util::AbstractCollection< URI >** (p. 151), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 151), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 151), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 151), **decaf::util::AbstractCollection< Pointer<**

Command > > (p. 151), **decaf::util::AbstractCollection**< **Pointer**<
BackupTransport > > (p. 151), **decaf::util::AbstractCollection**<
cms::MessageProducer * > (p. 151), **decaf::util::AbstractCollection**<
cms::Destination * > (p. 151), **decaf::util::AbstractCollection**< **cms::Session**
* > (p. 151), **decaf::util::AbstractCollection**< **cms::Connection** * >
(p. 151), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **Mes-**
sageId >, **Pointer**< **Message** >, **MessageId::COMPARATOR** >
(p. 1149), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **Connec-**
tionId >, **Pointer**< **ConnectionState** >, **ConnectionId::COMPARATOR**
> (p. 1149), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **Con-**
sumerId >, **Pointer**< **ConsumerState** >, **ConsumerId::COMPARATOR**
> (p. 1149), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **SessionId**
>, **Pointer**< **SessionState** >, **SessionId::COMPARATOR** > (p. 1149),
decaf::util::concurrent::ConcurrentStlMap< **Pointer**< **LocalTransactionId** >,
Pointer< **TransactionState** >, **LocalTransactionId::COMPARATOR** > (p. 1149),
decaf::util::concurrent::ConcurrentStlMap< **Pointer**< **ProducerId** >, **Pointer**<
ProducerState >, **ProducerId::COMPARATOR** > (p. 1149), **decaf::util::StlMap**<
cms::Session *, **SessionResolver** * > (p. 3378), **decaf::util::StlMap**< **std::string**,
WireFormatFactory * > (p. 3378), **decaf::util::StlMap**< **std::string**, **PrimitiveVal-**
ueNode > (p. 3378), **decaf::util::StlMap**< **std::string**, **cms::Queue** * > (p. 3378),
decaf::util::StlMap< **Pointer**< **commands::ProducerId** >, **ActiveMQProducer** *, **com-**
mands::ProducerId::COMPARATOR > (p. 3378), **decaf::util::StlMap**< **std::string**,
CachedConsumer * > (p. 3378), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId**
>, **ActiveMQConsumer** *, **commands::ConsumerId::COMPARATOR** > (p. 3378),
decaf::util::StlMap< **std::string**, **TransportFactory** * > (p. 3378), **decaf::util::StlMap**<
Pointer< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR**
> (p. 3378), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 3378),
decaf::util::StlMap< **Pointer**< **commands::ConsumerId** >, **Dispatcher** *, **com-**
mands::ConsumerId::COMPARATOR > (p. 3378), **decaf::util::StlMap**< **std::string**,
CachedProducer * > (p. 3378), **decaf::util::StlMap**< **std::string**, **cms::Topic**
* > (p. 3378), **decaf::util::StlQueue**< **Pointer**< **Transport** > > (p. 3387),
decaf::util::StlQueue< **Pointer**< **MessageDispatch** > > (p. 3387), **decaf::util::StlQueue**<
Task > (p. 3387), **decaf::util::StlQueue**< **Pointer**< **Command** > > (p. 3387), and
decaf::util::StlQueue< **decaf::lang::Pointer**< **commands::MessageDispatch** > >
(p. 3387).

6.793.3.4 **virtual bool decaf::util::concurrent::Synchronizable::tryLock () throw (**
decaf::lang::exceptions::RuntimeException) [pure virtual]

Attempts to **Lock** (p. 2228) the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2436), **de-**
cafe::internal::util::concurrent::SynchronizableImpl (p. 3475), **decaf::io::InputStream**
(p. 1917), **decaf::io::OutputStream** (p. 2723), **decaf::util::AbstractCollection**< **E**

> (p. 154), decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1154), decaf::util::concurrent::Mutex (p. 2606), decaf::util::StlMap< K, V, COMPARATOR > (p. 3380), decaf::util::StlQueue< T > (p. 3388), decaf::util::AbstractCollection< transport::TransportListener * > (p. 154), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 154), decaf::util::AbstractCollection< Resource * > (p. 154), decaf::util::AbstractCollection< cms::MessageConsumer * > (p. 154), decaf::util::AbstractCollection< CompositeTask * > (p. 154), decaf::util::AbstractCollection< URI > (p. 154), decaf::util::AbstractCollection< ActiveMQSession * > (p. 154), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 154), decaf::util::AbstractCollection< PrimitiveValueNode > > (p. 154), decaf::util::AbstractCollection< Pointer< Command > > (p. 154), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 154), decaf::util::AbstractCollection< cms::MessageProducer * > > (p. 154), decaf::util::AbstractCollection< cms::Destination * > > (p. 154), decaf::util::AbstractCollection< cms::Session * > > (p. 154), decaf::util::AbstractCollection< cms::Connection * > > (p. 154), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > > (p. 1154), decaf::util::StlMap< cms::Session *, SessionResolver * > > (p. 3380), decaf::util::StlMap< std::string, WireFormatFactory * > > (p. 3380), decaf::util::StlMap< std::string, PrimitiveValueNode > > (p. 3380), decaf::util::StlMap< std::string, cms::Queue * > > (p. 3380), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > > (p. 3380), decaf::util::StlMap< std::string, CachedConsumer * > > (p. 3380), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > > (p. 3380), decaf::util::StlMap< std::string, TransportFactory * > > (p. 3380), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > > (p. 3380), decaf::util::StlMap< int, Pointer< Command > > > (p. 3380), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > > (p. 3380), decaf::util::StlMap< std::string, CachedProducer * > > (p. 3380), decaf::util::StlMap< std::string, cms::Topic * > > (p. 3380), decaf::util::StlQueue< Pointer< Transport > > > (p. 3388), decaf::util::StlQueue< Pointer< MessageDispatch > > > (p. 3388), decaf::util::StlQueue< Task > > (p. 3388), decaf::util::StlQueue< Pointer< Command > > > (p. 3388), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > > (p. 3388).

6.793.3.5 virtual void decaf::util::concurrent::Synchronizable::unlock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2436), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3475), `decaf::io::InputStream` (p. 1918), `decaf::io::OutputStream` (p. 2723), `decaf::util::AbstractCollection< E >` (p. 154), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1154), `decaf::util::concurrent::Mutex` (p. 2607), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3380), `decaf::util::StlQueue< T >` (p. 3388), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 154), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 154), `decaf::util::AbstractCollection< Resource * >` (p. 154), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 154), `decaf::util::AbstractCollection< CompositeTask * >` (p. 154), `decaf::util::AbstractCollection< URI >` (p. 154), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 154), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 154), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 154), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 154), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 154), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 154), `decaf::util::AbstractCollection< cms::Destination * >` (p. 154), `decaf::util::AbstractCollection< cms::Session * >` (p. 154), `decaf::util::AbstractCollection< cms::Connection * >` (p. 154), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1154), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1154), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1154), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1154), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1154), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1154), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3380), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3380), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3380), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3380), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3380), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3380), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3380), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3380), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3380), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3380), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3380), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3380), `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3380), `decaf::util::StlQueue< Pointer< Transport > >` (p. 3388), `decaf::util::StlQueue< Pointer< MessageDispatch > >` (p. 3388), `decaf::util::StlQueue< Task >` (p. 3388), `decaf::util::StlQueue< Pointer< Command > >` (p. 3388), and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >` (p. 3388).

6.793.3.6 virtual void decaf::util::concurrent::Synchronizable::wait
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException,
 decaf::lang::exceptions::InterruptedException) [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2437), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 3475), **decaf::io::InputStream** (p. 1918), **decaf::io::OutputStream** (p. 2723), **decaf::util::AbstractCollection< E >** (p. 154), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1155), **decaf::util::concurrent::Mutex** (p. 2607), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3380), **decaf::util::StlQueue< T >** (p. 3388), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 154), **decaf::util::AbstractCollection< Pointer< Synchroniza- tion > >** (p. 154), **decaf::util::AbstractCollection< Resource * >** (p. 154), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 154), **decaf::util::AbstractCollection< CompositeTask * >** (p. 154), **decaf::util::AbstractCollection< URI >** (p. 154), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 154), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 154), **decaf::util::AbstractCollection< Prim- itiveValueNode >** (p. 154), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 154), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 154), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 154), **decaf::util::AbstractCollection< cms::Destination * >** (p. 154), **decaf::util::AbstractCollection< cms::Session * >** (p. 154), **decaf::util::AbstractCollection< cms::Connection * >** (p. 154), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Mes- sageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1155), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec- tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1155), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Con- sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1155), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1155), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1155), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1155), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 3380), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 3380), **decaf::util::StlMap< std::string, PrimitiveVal- ueNode >** (p. 3380), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 3380), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com- mands::ProducerId::COMPARATOR >** (p. 3380), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 3380), **decaf::util::StlMap< Pointer< commands::ConsumerId**

>, `ActiveMQConsumer *`, `commands::ConsumerId::COMPARATOR` > (p. 3380), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3380), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3380), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3380), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3380), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3380), `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3380), `decaf::util::StlQueue< Pointer< Transport > >` (p. 3388), `decaf::util::StlQueue< Pointer< MessageDispatch > >` (p. 3388), `decaf::util::StlQueue< Task >` (p. 3388), `decaf::util::StlQueue< Pointer< Command > >` (p. 3388), and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >` (p. 3388).

6.793.3.7 `virtual void decaf::util::concurrent::Synchronizable::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)` [pure virtual]

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or `WAIT_INFINITE`

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2436), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3476), `decaf::io::InputStream` (p. 1918), `decaf::io::OutputStream` (p. 2723), `decaf::util::AbstractCollection< E >` (p. 155), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1154), `decaf::util::concurrent::Mutex` (p. 2608), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3381), `decaf::util::StlQueue< T >` (p. 3389), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 155), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 155), `decaf::util::AbstractCollection< Resource * >` (p. 155), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 155), `decaf::util::AbstractCollection< CompositeTask * >` (p. 155), `decaf::util::AbstractCollection< URI >` (p. 155), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 155), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 155), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 155), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 155), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 155), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 155), `decaf::util::AbstractCollection< cms::Destination * >` (p. 155), `decaf::util::AbstractCollection< cms::Session * >` (p. 155), `decaf::util::AbstractCollection< cms::Connection * >`

(p. 155), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1154), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1154), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3381), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3381), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3381), decaf::util::StlMap< std::string, cms::Queue * > (p. 3381), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3381), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3381), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3381), decaf::util::StlMap< std::string, TransportFactory * > (p. 3381), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3381), decaf::util::StlMap< int, Pointer< Command > > (p. 3381), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3381), decaf::util::StlMap< std::string, CachedProducer * > (p. 3381), decaf::util::StlMap< std::string, cms::Topic * > (p. 3381), decaf::util::StlQueue< Pointer< Transport > > (p. 3389), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3389), decaf::util::StlQueue< Task > (p. 3389), decaf::util::StlQueue< Pointer< Command > > (p. 3389), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3389).

6.793.3.8 virtual void decaf::util::concurrent::Synchronizable::wait (long long *milliseconds*, int *nanos*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of

[0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3461) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2437), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 3476), **decaf::io::InputStream** (p. 1919), **decaf::io::OutputStream** (p. 2724), **decaf::util::AbstractCollection< E >** (p. 154), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1155), **decaf::util::concurrent::Mutex** (p. 2608), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3381), **decaf::util::StlQueue< T >** (p. 3389), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 154), **decaf::util::AbstractCollection< Pointer< Synchroniza- tion > >** (p. 154), **decaf::util::AbstractCollection< Resource * >** (p. 154), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 154), **decaf::util::AbstractCollection< CompositeTask * >** (p. 154), **decaf::util::AbstractCollection< URI >** (p. 154), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 154), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 154), **decaf::util::AbstractCollection< Prim- itiveValueNode >** (p. 154), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 154), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 154), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 154), **decaf::util::AbstractCollection< cms::Destination * >** (p. 154), **decaf::util::AbstractCollection< cms::Session * >** (p. 154), **decaf::util::AbstractCollection< cms::Connection * >** (p. 154), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Mes- sageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1155), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec- tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1155), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Con- sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1155), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1155), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1155), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1155), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 3381), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 3381), **decaf::util::StlMap< std::string, PrimitiveVal- ueNode >** (p. 3381), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 3381), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com- mands::ProducerId::COMPARATOR >** (p. 3381), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 3381), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 3381), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 3381), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3381), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3381), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, com- mands::ConsumerId::COMPARATOR >** (p. 3381), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 3381), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 3381), **decaf::util::StlQueue< Pointer< Transport > >** (p. 3389), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 3389), **decaf::util::StlQueue<**

Task > (p. 3389), **decaf::util::StlQueue< Pointer< Command > >** (p. 3389), and **decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >** (p. 3389).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Synchronizable.h`

6.794 decaf::internal::util::concurrent::SynchronizableImpl Class Reference

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

```
#include <src/main/decaf/internal/util/concurrent/SynchronizableImpl.h>
```

Inheritance diagram for `decaf::internal::util::concurrent::SynchronizableImpl`:

Public Member Functions

- **SynchronizableImpl** ()
- virtual **~SynchronizableImpl** ()
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.794.1 Detailed Description

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

Since

1.0

6.794.2 Constructor & Destructor Documentation

6.794.2.1 decaf::internal::util::concurrent::SynchronizableImpl::SynchronizableImpl ()

6.794.2.2 virtual decaf::internal::util::concurrent::SynchronizableImpl::~~SynchronizableImpl () [virtual]

6.794.3 Member Function Documentation

6.794.3.1 virtual void decaf::internal::util::concurrent::SynchronizableImpl::lock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements decaf::util::concurrent::Synchronizable (p. 3463).

6.794.3.2 virtual void decaf::internal::util::concurrent::SynchronizableImpl::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3464).

6.794.3.3 virtual void decaf::internal::util::concurrent::SynchronizableImpl::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3465).

6.794.3.4 virtual bool decaf::internal::util::concurrent::SynchronizableImpl::tryLock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3466).

6.794.3.5 virtual void decaf::internal::util::concurrent::SynchronizableImpl::unlock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3467).

6.794.3.6 virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3468).

6.794.3.7 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3470).

6.794.3.8 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3471).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/SynchronizableImpl.h`

6.795 activemq::core::Synchronization Class Reference

Transacted Object **Synchronization** (p. 3477), used to sync the events of a Transaction with the items in the Transaction.

```
#include <src/main/activemq/core/Synchronization.h>
```

Public Member Functions

- virtual **~Synchronization** ()
- virtual void **beforeEnd** ()=0 throw (exceptions::ActiveMQException)
- virtual void **afterCommit** ()=0 throw (exceptions::ActiveMQException)
- virtual void **afterRollback** ()=0 throw (exceptions::ActiveMQException)

6.795.1 Detailed Description

Transacted Object **Synchronization** (p. 3477), used to sync the events of a Transaction with the items in the Transaction.

6.795.2 Constructor & Destructor Documentation

6.795.2.1 `virtual activemq::core::Synchronization::~~Synchronization () [inline, virtual]`

6.795.3 Member Function Documentation

6.795.3.1 `virtual void activemq::core::Synchronization::afterCommit () throw (exceptions::ActiveMQException) [pure virtual]`

6.795.3.2 `virtual void activemq::core::Synchronization::afterRollback () throw (exceptions::ActiveMQException) [pure virtual]`

6.795.3.3 `virtual void activemq::core::Synchronization::beforeEnd () throw (exceptions::ActiveMQException) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/Synchronization.h`

6.796 decaf::util::concurrent::SynchronousQueue< E > Class Template Reference

A **blocking queue** (p. 774) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

```
#include <src/main/decaf/util/concurrent/SynchronousQueue.h>
```

Inheritance diagram for `decaf::util::concurrent::SynchronousQueue< E >`:

Data Structures

- class **EmptyIterator**

Public Member Functions

- **SynchronousQueue** ()
- virtual **~SynchronousQueue** ()
- virtual void **put** (const E &value) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)

Adds the specified element to this queue, waiting if necessary for another thread to receive it.

- virtual bool **offer** (const E &e, long timeout, const **TimeUnit** &unit) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)

Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.

- virtual bool **offer** (const E &value) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Inserts the specified element into this queue, if another thread is waiting to receive it.
- virtual E **take** () throw (decaf::lang::exceptions::InterruptedException)
Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.
- virtual bool **poll** (E &result, long long timeout, const TimeUnit &unit) throw (decaf::lang::exceptions::InterruptedException)
Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.
- virtual bool **poll** (E &result)
Retrieves and removes the head of this queue, if another thread is currently making an element available.
- virtual bool **equals** (const Collection< E > &value) const
*Answers true if this **Collection** (p. 1097) and the one given are the same size and if each element contained in the **Collection** (p. 1097) given is equal to an element contained in this collection.*
- virtual decaf::util::Iterator< E > * **iterator** ()
- virtual decaf::util::Iterator< E > * **iterator** () const
- virtual bool **isEmpty** () const
Returns true if this collection contains no elements.
- virtual std::size_t **size** () const
Returns the number of elements in this collection.
- virtual int **remainingCapacity** () const
*Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or **Integer::MAX_VALUE** if there is no intrinsic limit.*
- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
Removes all elements of the queue.
- virtual bool **contains** (const E &value DECAF_UNUSED) const throw (lang::Exception)
- virtual bool **containsAll** (const Collection< E > &collection) const throw (lang::Exception)
Returns true if this collection contains all of the elements in the specified collection.
- virtual bool **remove** (const E &value DECAF_UNUSED) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
- virtual bool **removeAll** (const Collection< E > &collection DECAF_UNUSED) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
- virtual bool **retainAll** (const Collection< E > &collection DECAF_UNUSED) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

- virtual bool **peek** (E &result DECAF_UNUSED) const
- virtual std::vector< E > **toArray** () const

*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1097).*

- virtual std::size_t **drainTo** (**Collection**< E > &c) throw
(decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)

Removes all available elements from this queue and adds them to the given collection.

- virtual std::size_t **drainTo** (**Collection**< E > &c, std::size_t maxElements) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)

Removes at most the given number of available elements from this queue and adds them to the given collection.

6.796.1 Detailed Description

template<typename E> class decaf::util::concurrent::SynchronousQueue< E >

A **blocking queue** (p. 774) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa. A synchronous queue does not have any internal capacity, not even a capacity of one. You cannot **peek** at a synchronous queue because an element is only present when you try to remove it; you cannot insert an element (using any method) unless another thread is trying to remove it; you cannot iterate as there is nothing to iterate. The *head* of the queue is the element that the first queued inserting thread is trying to add to the queue; if there is no such queued thread then no element is available for removal and **poll()** (p. 3484) will return **null**. For purposes of other **Collection** (p. 1097) methods (for example **contains**), a **SynchronousQueue** (p. 3477) acts as an empty collection. This queue does not permit **null** elements.

Synchronous queues are similar to rendezvous channels used in CSP and Ada. They are well suited for handoff designs, in which an object running in one thread must sync up with an object running in another thread in order to hand it some information, event, or task.

This class supports an optional fairness policy for ordering waiting producer and consumer threads. By default, this ordering is not guaranteed. However, a queue constructed with fairness set to **true** grants threads access in FIFO order.

This class and its iterator implement all of the *optional* methods of the **Collection** (p. 1097) and **Iterator** (p. 2012) interfaces.

Since

1.0

6.796.2 Constructor & Destructor Documentation

6.796.2.1 `template<typename E > decaf::util::concurrent::SynchronousQueue< E >::SynchronousQueue () [inline]`

6.796.2.2 `template<typename E > virtual decaf::util::concurrent::SynchronousQueue< E >::~~SynchronousQueue () [inline, virtual]`

6.796.3 Member Function Documentation

6.796.3.1 `template<typename E > virtual void decaf::util::concurrent::SynchronousQueue< E >::clear () throw (lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes all elements of the queue.

This implementation repeatedly invokes poll until it returns the empty marker.

Reimplemented from `decaf::util::AbstractQueue< E >` (p. 160).

6.796.3.2 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::contains (const E &value DECAF_UNUSED) const throw (lang::Exception) [inline, virtual]`

6.796.3.3 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::containsAll (const Collection< E > & collection) const throw (lang::Exception) [inline, virtual]`

Returns true if this collection contains all of the elements in the specified collection.

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

Parameters

collection collection to be checked for containment in this collection

Returns

true if this collection contains all of the elements in the specified collection.

Exceptions

Exception if an error occurs,

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 149).

```

6.796.3.4  template<typename E > virtual std::size_t
             decaf::util::concurrent::SynchronousQueue< E
             >::drainTo ( Collection< E > & c ) throw (
             decaf::lang::exceptions::UnsupportedOperationException,
             decaf::lang::exceptions::IllegalArgumentException ) [inline, virtual]

```

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

c the collection to transfer elements into

Returns

the number of elements transferred

Exceptions

UnsupportedOperationException if addition of elements is not supported by the specified collection

IllegalArgumentException if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements `decaf::util::concurrent::BlockingQueue< E >` (p. 777).

References `decaf::util::concurrent::SynchronousQueue< E >::poll()`.

```

6.796.3.5  template<typename E > virtual std::size_t
             decaf::util::concurrent::SynchronousQueue< E >::drainTo
             ( Collection< E > & c, std::size_t maxElements ) throw
             ( decaf::lang::exceptions::UnsupportedOperationException,
             decaf::lang::exceptions::IllegalArgumentException ) [inline, virtual]

```

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

c the collection to transfer elements into

maxElements the maximum number of elements to transfer

Returns

the number of elements transferred

Exceptions

UnsupportedOperationException if addition of elements is not supported by the specified collection

IllegalArgumentException if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 778).

References **decaf::util::concurrent::SynchronousQueue< E >::poll()**.

6.796.3.6 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::equals (
const Collection< E > & collection) const [inline, virtual]`

Answers true if this **Collection** (p. 1097) and the one given are the same size and if each element contained in the **Collection** (p. 1097) given is equal to an element contained in this collection.

Parameters

collection - The **Collection** (p. 1097) to be compared to this one.

Returns

true if this **Collection** (p. 1097) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 149).

6.796.3.7 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::isEmpty
() const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 3486) == 0.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 150).

- 6.796.3.8** `template<typename E > virtual decaf::util::Iterator<E>*`
`decaf::util::concurrent::SynchronousQueue< E >::iterator () const`
`[inline, virtual]`
- 6.796.3.9** `template<typename E > virtual decaf::util::Iterator<E>*`
`decaf::util::concurrent::SynchronousQueue< E >::iterator () [inline,`
`virtual]`
- 6.796.3.10** `template<typename E > virtual bool`
`decaf::util::concurrent::SynchronousQueue< E >::offer (`
`const E & e, long timeout, const TimeUnit & unit`
`) throw (decaf::lang::exceptions::InterruptedException,`
`decaf::lang::exceptions::NullPointerException, de-`
`caf::lang::exceptions::IllegalArgumentException) [inline,`
`virtual]`

Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.

Returns

`true` if successful, or `false` if the specified waiting time elapses before a consumer appears.

Exceptions

- InterruptedException*** Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
- NullPointerException*** Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
- IllegalArgumentException*** Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Implements `decaf::util::concurrent::BlockingQueue< E >` (p. 779).

- 6.796.3.11** `template<typename E > virtual bool`
`decaf::util::concurrent::SynchronousQueue< E >::offer (const E`
`& value) throw (decaf::lang::exceptions::NullPointerException,`
`decaf::lang::exceptions::IllegalArgumentException) [inline, virtual]`

Inserts the specified element into this queue, if another thread is waiting to receive it.

Parameters

value the element to add to the **Queue** (p. 2948)

Returns

`true` if the element was added to this queue, else `false`

Exceptions

- NullPointerException*** if the **Queue** (p. 2948) implementation does not allow Null values to be inserted into the **Queue** (p. 2948).

IllegalArgumentException if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 2949).

6.796.3.12 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::peek (
E &result DECAF_UNUSED) const [inline, virtual]`

6.796.3.13 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::poll (
E & result, long long timeout, const TimeUnit & unit) throw (
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.

Parameters

result a reference to the value where the head of the **Queue** (p. 2948) should be copied to.

timeout the time that the method should block if there is no element available to return.

unit the Time Units that the timeout value represents.

Returns

true if the head of the **Queue** (p. 2948) was copied to the result param or false if no value could be returned.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 779).

Referenced by **decaf::util::concurrent::SynchronousQueue< E >::drainTo()**.

6.796.3.14 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::poll (
E & result) [inline, virtual]`

Retrieves and removes the head of this queue, if another thread is currently making an element available.

Parameters

result a reference to the value where the head of the **Queue** (p. 2948) should be copied to.

Returns

true if the head of the **Queue** (p. 2948) was copied to the result param or false if no value could be returned.

Implements **decaf::util::Queue< E >** (p. 2950).

```

6.796.3.15  template<typename E > virtual void
               decaf::util::concurrent::SynchronousQueue< E >::put (  const E
               &  value ) throw ( decaf::lang::exceptions::InterruptedException,
               decaf::lang::exceptions::NullPointerException, de-
               ccaf::lang::exceptions::IllegalArgumentException ) [inline,
               virtual]

```

Adds the specified element to this queue, waiting if necessary for another thread to receive it.

Parameters

value the element to add to the **Queue** (p. 2948).

Exceptions

InterruptedException Inserts the specified element into this queue, waiting if necessary for space to become available.

NullPointerException Inserts the specified element into this queue, waiting if necessary for space to become available.

IllegalArgumentException Inserts the specified element into this queue, waiting if necessary for space to become available.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 780).

```

6.796.3.16  template<typename E > virtual int
               decaf::util::concurrent::SynchronousQueue< E
               >::remainingCapacity (  ) const [inline, virtual]

```

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or **Integer::MAX_VALUE** if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting **remainingCapacity** because it may be the case that another thread is about to insert or remove an element.

Returns

the remaining capacity

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 780).

- 6.796.3.17** `template<typename E> virtual bool
decaf::util::concurrent::SynchronousQueue< E >::remove
(const E &value DECAF_UNUSED) throw (
lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException) [inline, virtual]`
- 6.796.3.18** `template<typename E> virtual bool
decaf::util::concurrent::SynchronousQueue< E
>::removeAll (const Collection< E > &collection DECAF_UNUSED
) throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException) [inline, virtual]`
- 6.796.3.19** `template<typename E> virtual bool
decaf::util::concurrent::SynchronousQueue< E
>::retainAll (const Collection< E > &collection DECAF_UNUSED
) throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException) [inline, virtual]`
- 6.796.3.20** `template<typename E> virtual std::size_t
decaf::util::concurrent::SynchronousQueue< E >::size () const
[inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX_VALUE elements, returns Integer.MAX_VALUE.

Returns

the number of elements in this collection

Implements `decaf::util::Collection< E >` (p. 1106).

- 6.796.3.21** `template<typename E> virtual E
decaf::util::concurrent::SynchronousQueue< E >::take (
) throw (decaf::lang::exceptions::InterruptedException) [inline,
virtual]`

Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.

Returns

the head of this queue

Exceptions

InterruptedException Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

Implements `decaf::util::concurrent::BlockingQueue< E >` (p. 780).

- 6.796.3.22** `template<typename E> virtual std::vector<E>
decaf::util::concurrent::SynchronousQueue< E >::toArray () const
[inline, virtual]`

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1097).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

Returns

an vector of copies of all the elements from this **Collection** (p. 1097)

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 153).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/SynchronousQueue.h`

6.797 decaf::lang::System Class Reference

The **System** (p. 3487) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

```
#include <src/main/decaf/lang/System.h>
```

Public Member Functions

- `virtual ~System ()`

Static Public Member Functions

- static void **arraycopy** (const unsigned char *src, std::size_t srcPos, unsigned char *dest, std::size_t destPos, std::size_t length)
Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.
- static void **arraycopy** (const short *src, std::size_t srcPos, short *dest, std::size_t destPos, std::size_t length)
Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.
- static void **arraycopy** (const int *src, std::size_t srcPos, int *dest, std::size_t destPos, std::size_t length)
Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.
- static void **arraycopy** (const long long *src, std::size_t srcPos, long long *dest, std::size_t destPos, std::size_t length)
Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- static const **util::Map**< std::string, std::string > & **getenv** ()
Enumerates the system environment and returns a map of env variable names to the string values they hold.
- static std::string **getenv** (const std::string &name)
Reads an environment value from the system and returns it as a string object.
- static void **unsetenv** (const std::string &name)
Clears a set environment value if one is set.
- static void **setenv** (const std::string &name, const std::string &value)
Sets the specified system property to the value given.
- static long long **currentTimeMillis** ()
Returns the current time in milliseconds.
- static long long **nanoTime** ()
Returns the current value of the most precise available system timer, in nanoseconds.
- static int **availableProcessors** ()
Returns the number of processors available for execution of Decaf Threads.
- static **decaf::util::Properties** & **getProperties** ()
Gets the Properties object that holds the Properties accessed from calls to getProperty and setProperty.
- static std::string **getProperty** (const std::string &key)
*Gets the specified **System** (p. 3487) property if set, otherwise returns an empty string.*
- static std::string **getProperty** (const std::string &key, const std::string &defaultValue)
*Gets the specified **System** (p. 3487) property if set, otherwise returns the specified default value.*
- static std::string **setProperty** (const std::string &key, const std::string &value)
*Sets the **System** (p. 3487) Property to the specified value.*
- static std::string **clearProperty** (const std::string &key)
Clear any value associated with the system property specified.

Protected Member Functions

- **System** ()

Friends

- class **decaf::lang::Runtime**

6.797.1 Detailed Description

The **System** (p.3487) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

Since

1.0

6.797.2 Constructor & Destructor Documentation

6.797.2.1 `decaf::lang::System::System ()` [protected]

6.797.2.2 `virtual decaf::lang::System::~~System ()` [inline, virtual]

6.797.3 Member Function Documentation

6.797.3.1 `static void decaf::lang::System::arraycopy (const unsigned char * src, std::size_t srcPos, unsigned char * dest, std::size_t destPos, std::size_t length)` [static]

Copies the number of elements specified by *length* from the source array starting at the given source offset specified by *srcPos* to the dest array starting at the given destination offset given by *destPos*.

Parameters

src The source array to copy from.

srcPos The position in the array to start copying from.

dest The destination array to copy to.

destPos The position in the destination array to start writing at.

length The number of elements to copy from *src* to *dest*.

Exceptions

NullPointerException if *src* or *dest* are NULL.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::clone()`.

6.797.3.2 `static void decaf::lang::System::arraycopy (const short * src, std::size_t srcPos, short * dest, std::size_t destPos, std::size_t length)` [static]

Copies the number of elements specified by *length* from the source array starting at the given source offset specified by *srcPos* to the dest array starting at the given destination offset given by *destPos*.

Parameters

src The source array to copy from.

srcPos The position in the array to start copying from.

dest The destination array to copy to.

destPos The position in the destination array to start writing at.

length The number of elements to copy from src to dest.

Exceptions

NullPointerException if src or dest are NULL.

6.797.3.3 `static void decaf::lang::System::arraycopy (const long long * src, std::size_t srcPos, long long * dest, std::size_t destPos, std::size_t length) [static]`

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

src The source array to copy from.

srcPos The position in the array to start copying from.

dest The destination array to copy to.

destPos The position in the destination array to start writing at.

length The number of elements to copy from src to dest.

Exceptions

NullPointerException if src or dest are NULL.

6.797.3.4 `static void decaf::lang::System::arraycopy (const int * src, std::size_t srcPos, int * dest, std::size_t destPos, std::size_t length) [static]`

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

src The source array to copy from.

srcPos The position in the array to start copying from.

dest The destination array to copy to.

destPos The position in the destination array to start writing at.

length The number of elements to copy from src to dest.

Exceptions

NullPointerException if src or dest are NULL.

6.797.3.5 `static int decaf::lang::System::availableProcessors () [static]`

Returns the number of processors available for execution of Decaf Threads.

This value may change during a particular execution of a Decaf based application. Applications that are sensitive to the number of available processors should therefore occasionally poll this property and adjust their resource usage appropriately.

Returns

the number of available processors.

6.797.3.6 `static std::string decaf::lang::System::clearProperty (const std::string &key) [static]`

Clear any value associated with the system property specified.

Parameters

key The key name of the system property to clear.

Returns

the previous value of the property named by key if there was one, otherwise returns an empty string.

Exceptions

IllegalArgumentException if key is an empty string.

6.797.3.7 `static long long decaf::lang::System::currentTimeMillis () [static]`

Returns the current time in milliseconds.

Note that while the unit of time of the return value is a millisecond, the granularity of the value depends on the underlying operating system and may be larger. For example, many operating systems measure time in units of tens of milliseconds.

See the description of the class Date for a discussion of slight discrepancies that may arise between "computer time" and coordinated universal time (UTC).

Returns

the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.

6.797.3.8 `static const util::Map<std::string, std::string>& decaf::lang::System::getenv () [static]`

Enumerates the system environment and returns a map of env variable names to the string values they hold.

Returns

A Map of all environment variables.

Exceptions

Exception (p. 1712) if an error occurs while getting the Environment Map.

6.797.3.9 `static std::string decaf::lang::System::getenv (const std::string & name) [static]`

Reads an environment value from the system and returns it as a string object.

Parameters

name The environment variable to read.

Returns

a string with the value from the variables or ""

Exceptions

an Exception (p. 1712) if an error occurs while reading the Env.

6.797.3.10 `static decaf::util::Properties& decaf::lang::System::getProperties () [static]`

Gets the Properties object that holds the Properties accessed from calls to getProperty and setProperty.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

Returns

a reference to the static system Properties object.

6.797.3.11 `static std::string decaf::lang::System::getProperty (const std::string & key) [static]`

Gets the specified **System** (p. 3487) property if set, otherwise returns an empty string.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

Parameters

key The key name of the desired system property to retrieve.

Returns

an empty string if the named property is not set, otherwise returns the value.

Exceptions

IllegalArgumentException if key is an empty string.

6.797.3.12 `static std::string decaf::lang::System::getProperty (const std::string & key, const std::string & defaultValue) [static]`

Gets the specified **System** (p. 3487) property if set, otherwise returns the specified default value.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

Parameters

key The key name of the desired system property to retrieve.

defaultValue The default value to return if the key is not set in the **System** (p. 3487) properties.

Returns

the value of the named system property or the defaultValue if the property isn't set..

Exceptions

IllegalArgumentException if key is an empty string.

6.797.3.13 `static long long decaf::lang::System::nanoTime () [static]`

Returns the current value of the most precise available system timer, in nanoseconds.

This method can only be used to measure elapsed time and is not related to any other notion of system or wall-clock time. The value returned represents nanoseconds since some fixed but arbitrary time (perhaps in the future, so values may be negative). This method provides nanosecond precision, but not necessarily nanosecond accuracy. No guarantees are made about how frequently values change. Differences in successive calls that span greater than approximately 292 years (263 nanoseconds) will not accurately compute elapsed time due to numerical overflow.

For example, to measure how long some code takes to execute:

```
long long startTime = System::nanoTime() (p. 3493); // ... the code being measured ... long
long estimatedTime = System::nanoTime() (p. 3493) - startTime;
```

Returns

The current value of the system timer, in nanoseconds.

6.797.3.14 `static void decaf::lang::System::setenv (const std::string & name, const std::string & value) [static]`

Sets the specified system property to the value given.

Parameters

name The name of the environment variables to set.

value The value to assign to name.

Exceptions

an Exception (p. 1712) if an error occurs when setting the environment variable.

6.797.3.15 `static std::string decaf::lang::System::setProperty (const std::string & key, const std::string & value) [static]`

Sets the **System** (p.3487) Property to the specified value.

Parameters

key The key name of the system property to set to the given value.

value The value to assign to the key.

Returns

the previous value of the property named by key if there was one, otherwise returns an empty string.

Exceptions

IllegalArgumentException if key is an empty string.

6.797.3.16 `static void decaf::lang::System::unsetenv (const std::string & name) [static]`

Clears a set environment value if one is set.

Parameters

name The environment variables to clear.

Exceptions

an Exception (p.1712) if an error occurs while reading the environment.

6.797.4 Friends And Related Function Documentation

6.797.4.1 `friend class decaf::lang::Runtime [friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/System.h`

6.798 activemq::threads::Task Class Reference

Represents a unit of work that requires one or more iterations to complete.

`#include <src/main/activemq/threads/Task.h>`

Inheritance diagram for `activemq::threads::Task`:

Public Member Functions

- virtual `~Task()`
- virtual bool `iterate()`=0

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

6.798.1 Detailed Description

Represents a unit of work that requires one or more iterations to complete.

Since

3.0

6.798.2 Constructor & Destructor Documentation

6.798.2.1 virtual `activemq::threads::Task::~Task()` [inline, virtual]

6.798.3 Member Function Documentation

6.798.3.1 virtual bool `activemq::threads::Task::iterate()` [pure virtual]

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

Returns

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implemented in `activemq::core::ActiveMQSessionExecutor` (p. 485), `activemq::transport::failover::BackupTransportPool` (p. 694), `activemq::transport::failover::CloseTransportsTask` (p. 1067), and `activemq::transport::failover::FailoverTransport` (p. 1758).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/Task.h`

6.799 decaf::util::concurrent::TaskListener Class Reference

```
#include <src/main/decaf/util/concurrent/TaskListener.h>
```

Public Member Functions

- virtual `~TaskListener()`
- virtual void `onTaskComplete(lang::Runnable *task)`=0

Called when a queued task has completed, the task that finished is passed along for user consumption.

- virtual void **onTaskException** (lang::Runnable *task, lang::Exception &ex)=0

Called when a queued task has thrown an exception while being run.

6.799.1 Constructor & Destructor Documentation

- 6.799.1.1** virtual decaf::util::concurrent::TaskListener::~~TaskListener ()
[inline, virtual]

6.799.2 Member Function Documentation

- 6.799.2.1** virtual void decaf::util::concurrent::TaskListener::onTaskComplete (lang::Runnable * *task*) [pure virtual]

Called when a queued task has completed, the task that finished is passed along for user consumption.

Parameters

task Runnable Pointer to the task that finished

- 6.799.2.2** virtual void decaf::util::concurrent::TaskListener::onTaskException (lang::Runnable * *task*, lang::Exception & *ex*) [pure virtual]

Called when a queued task has thrown an exception while being run.

The Callee should assume that this was an unrecoverable exception and that this task is now defunct.

Parameters

task Runnable Pointer to the task

ex The ActiveMQException that was thrown.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**TaskListener.h**

6.800 activemq::threads::TaskRunner Class Reference

```
#include <src/main/activemq/threads/TaskRunner.h>
```

Inheritance diagram for activemq::threads::TaskRunner:

Public Member Functions

- virtual ~TaskRunner ()
- virtual void **shutdown** (unsigned int timeout)=0

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

- virtual void **shutdown** ()=0

Shutdown once the task has finished and the TaskRunner's thread has exited.

- virtual void **wakeup** ()=0

*Signal the **TaskRunner** (p. 3496) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3494) instance will be run until its iterate method has returned false indicating it is done.*

6.800.1 Constructor & Destructor Documentation

- 6.800.1.1 virtual **activemq::threads::TaskRunner::~TaskRunner** () [inline, virtual]

6.800.2 Member Function Documentation

- 6.800.2.1 virtual void **activemq::threads::TaskRunner::shutdown** (unsigned int *timeout*) [pure virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters

timeout - Time in Milliseconds to wait for the task to stop.

- 6.800.2.2 virtual void **activemq::threads::TaskRunner::shutdown** () [pure virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

- 6.800.2.3 virtual void **activemq::threads::TaskRunner::wakeup** () [pure virtual]

Signal the **TaskRunner** (p. 3496) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3494) instance will be run until its iterate method has returned false indicating it is done.

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**TaskRunner.h**

6.801 decaf::internal::net::tcp::TcpSocket Class Reference

Platform-independent implementation of the socket interface.

```
#include <src/main/decaf/internal/net/tcp/TcpSocket.h>
```

Inheritance diagram for decaf::internal::net::tcp::TcpSocket:

Public Member Functions

- **TcpSocket** () throw (decaf::net::SocketException)
Construct a non-connected socket.
- virtual ~**TcpSocket** ()
Releases the socket handle but not gracefully shut down the connection.
- SocketHandle **getSocketHandle** ()
Gets the handle for the socket.
- bool **isConnected** () const
- bool **isClosed** () const
- virtual std::string **getLocalAddress** () const
*Gets the value of the local Inet address the **Socket** (p. 3281) is bound to if bound, otherwise return the **InetAddress** (p. 1884) ANY value "0.0.0.0".*
Returns
the local address bound to, or ANY.
- virtual void **create** () throw (decaf::io::IOException)
*Creates the underlying platform **Socket** (p. 3281) data structures which allows for **Socket** (p. 3281) options to be applied.*
The created socket is in an unconnected state.
Exceptions
***IOException** if an I/O error occurs while attempting this operation.*
- virtual void **accept** (SocketImpl *socket) throw (decaf::io::IOException)
- virtual void **bind** (const std::string &ipaddress, int **port**) throw (decaf::io::IOException)
*Binds this **Socket** (p. 3281) instance to the local ip address and port number given.*
Parameters
***ipaddress** The address of local ip to bind to.*
***port** The port number on the host to bind to.*
Exceptions
***IOException** if an I/O error occurs while attempting this operation.*
- virtual void **connect** (const std::string &hostname, int **port**, int timeout) throw (decaf::io::IOException, decaf::net::SocketException, decaf::lang::exceptions::IllegalArgumentException)
Connects this socket to the given host and port.
Parameters
***hostname** The name of the host to connect to, or IP address.*
***port** The port number on the host to connect to.*
***timeout** Time in milliseconds to wait for a connection, 0 indicates forever.*
Exceptions
***IOException** if an I/O error occurs while attempting this operation.*
***SocketTimeoutException** (p. 3319) if the connect call times out due to timeout being set.*
***IllegalArgumentException** if a parameter has an illegal value.*

- virtual void **listen** (int backlog) throw (decaf::io::IOException)
Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.
If a connection indication arrives when the queue is full, the connection is refused.
Parameters
backlog The maximum length of the connection queue.
Exceptions
IOException if an I/O error occurs while attempting this operation.

- virtual decaf::io::InputStream * **getInputStream** () throw (decaf::io::IOException)
*Gets the InputStream linked to this **Socket** (p. 3281).*
Returns
*an InputStream pointer owned by the **Socket** (p. 3281) object.*
Exceptions
IOException if an I/O error occurs while attempting this operation.

- virtual decaf::io::OutputStream * **getOutputStream** () throw (decaf::io::IOException)
*Gets the OutputStream linked to this **Socket** (p. 3281).*
Returns
*an OutputStream pointer owned by the **Socket** (p. 3281) object.*
Exceptions
IOException if an I/O error occurs while attempting this operation.

- virtual int **available** () throw (decaf::io::IOException)
*Gets the number of bytes that can be read from the **Socket** (p. 3281) without blocking.*
Returns
*the number of bytes that can be read from the **Socket** (p. 3281) without blocking.*
Exceptions
IOException if an I/O error occurs while attempting this operation.

- virtual void **close** () throw (decaf::io::IOException)
Closes the socket, terminating any blocked reads or writes.
Exceptions
IOException if an I/O error occurs while attempting this operation.

- virtual void **shutdownInput** () throw (decaf::io::IOException)
Places the input stream for this socket at "end of stream".
*Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 3312) on the socket, the stream will return EOF.*
Exceptions
IOException if an I/O error occurs while attempting this operation.

- virtual void **shutdownOutput** () throw (decaf::io::IOException)

Disables the output stream for this socket.

*For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 3313) on the socket, the stream will throw an *IOException*.*

Exceptions

***IOException** if an I/O error occurs while attempting this operation.*

- virtual int **getOption** (int option) const throw (decaf::io::IOException)

*Gets the specified **Socket** (p. 3281) option.*

Parameters

***option** The **Socket** (p. 3281) options whose value is to be retrieved.*

Returns

the value of the given socket option.

Exceptions

***IOException** if an I/O error occurs while performing this operation.*

- virtual void **setOption** (int option, int value) throw (decaf::io::IOException)

*Sets the specified option on the **Socket** (p. 3281) if supported.*

Parameters

***option** The **Socket** (p. 3281) option to set.*

***value** The value of the socket option to apply to the socket.*

Exceptions

***IOException** if an I/O error occurs while performing this operation.*

- int **read** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

*Reads the requested data from the **Socket** and write it into the passed in buffer.*

- void **write** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

*Writes the specified data in the passed in buffer to the **Socket**.*

Protected Member Functions

- void **checkResult** (apr_status_t value) const throw (decaf::net::SocketException)

6.801.1 Detailed Description

Platform-independent implementation of the socket interface.

6.801.2 Constructor & Destructor Documentation

- 6.801.2.1** decaf::internal::net::tcp::TcpSocket::TcpSocket () throw (decaf::net::SocketException)

Construct a non-connected socket.

Exceptions

SocketException thrown if an error occurs while creating the Socket.

6.801.2.2 virtual decaf::internal::net::tcp::TcpSocket::~~TcpSocket () [virtual]

Releases the socket handle but not gracefully shut down the connection.

6.801.3 Member Function Documentation

6.801.3.1 virtual void decaf::internal::net::tcp::TcpSocket::accept (SocketImpl * *socket*) throw (decaf::io::IOException) [virtual]

6.801.3.2 virtual int decaf::internal::net::tcp::TcpSocket::available () throw (decaf::io::IOException) [virtual]

Gets the number of bytes that can be read from the **Socket** (p. 3281) without blocking.

Returns

the number of bytes that can be read from the **Socket** (p. 3281) without blocking.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 3308).

6.801.3.3 virtual void decaf::internal::net::tcp::TcpSocket::bind (const std::string & *ipaddress*, int *port*) throw (decaf::io::IOException) [virtual]

Binds this **Socket** (p. 3281) instance to the local ip address and port number given.

Parameters

ipaddress The address of local ip to bind to.

port The port number on the host to bind to.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 3308).

6.801.3.4 void decaf::internal::net::tcp::TcpSocket::checkResult (apr_status_t *value*) const throw (decaf::net::SocketException) [protected]

6.801.3.5 virtual void decaf::internal::net::tcp::TcpSocket::close () throw (decaf::io::IOException) [virtual]

Closes the socket, terminating any blocked reads or writes.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 3309).

6.801.3.6 `virtual void decaf::internal::net::tcp::TcpSocket::connect (const std::string & hostname, int port, int timeout) throw (decaf::io::IOException, decaf::net::SocketException, decaf::lang::exceptions::IllegalArgumentException) [virtual]`

Connects this socket to the given host and port.

Parameters

hostname The name of the host to connect to, or IP address.

port The port number on the host to connect to.

timeout Time in milliseconds to wait for a connection, 0 indicates forever.

Exceptions

IOException if an I/O error occurs while attempting this operation.

SocketTimeoutException (p. 3319) if the connect call times out due to timeout being set.

IllegalArgumentException if a parameter has an illegal value.

Implements **decaf::net::SocketImpl** (p. 3309).

6.801.3.7 `virtual void decaf::internal::net::tcp::TcpSocket::create () throw (decaf::io::IOException) [virtual]`

Creates the underlying platform **Socket** (p. 3281) data structures which allows for **Socket** (p. 3281) options to be applied.

The created socket is in an unconnected state.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 3309).

6.801.3.8 `virtual decaf::io::InputStream* decaf::internal::net::tcp::TcpSocket::getInputStream () throw (decaf::io::IOException) [virtual]`

Gets the InputStream linked to this **Socket** (p. 3281).

Returns

an InputStream pointer owned by the **Socket** (p. 3281) object.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 3310).

6.801.3.9 `virtual std::string decaf::internal::net::tcp::TcpSocket::getLocalAddress () const [virtual]`

Gets the value of the local Inet address the **Socket** (p. 3281) is bound to if bound, otherwise return the **InetAddress** (p. 1884) ANY value "0.0.0.0".

Returns

the local address bound to, or ANY.

Implements **decaf::net::SocketImpl** (p. 3310).

6.801.3.10 `virtual int decaf::internal::net::tcp::TcpSocket::getOption (int option) const throw (decaf::io::IOException) [virtual]`

Gets the specified **Socket** (p. 3281) option.

Parameters

option The **Socket** (p. 3281) options whose value is to be retrieved.

Returns

the value of the given socket option.

Exceptions

IOException if an I/O error occurs while performing this operation.

Implements **decaf::net::SocketImpl** (p. 3311).

6.801.3.11 `virtual decaf::io::OutputStream* decaf::internal::net::tcp::TcpSocket::getOutputStream () throw (decaf::io::IOException) [virtual]`

Gets the OutputStream linked to this **Socket** (p. 3281).

Returns

an OutputStream pointer owned by the **Socket** (p. 3281) object.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 3311).

6.801.3.12 `SocketHandle decaf::internal::net::tcp::TcpSocket::getSocketHandle () [inline]`

Gets the handle for the socket.

Returns

SocketHabler for this Socket, can be NULL

6.801.3.13 `bool decaf::internal::net::tcp::TcpSocket::isClosed () const [inline]`

Returns

true if the close method has been called on this Socket.

6.801.3.14 `bool decaf::internal::net::tcp::TcpSocket::isConnected () const [inline]`

Returns

true if the socketHandle is not in a disconnected state.

6.801.3.15 `virtual void decaf::internal::net::tcp::TcpSocket::listen (int backlog) throw (decaf::io::IOException) [virtual]`

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

Parameters

backlog The maximum length of the connection queue.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implements `decaf::net::SocketImpl` (p. 3311).

6.801.3.16 `int decaf::internal::net::tcp::TcpSocket::read (unsigned char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)`

Reads the requested data from the Socket and write it into the passed in buffer.

Parameters

buffer The buffer to read into

size The size of the specified buffer

offset The offset into the buffer where reading should start filling.

length The number of bytes past offset to fill with data.

Returns

the actual number of bytes read or -1 if at EOF.

Exceptions

IOException if an I/O error occurs during the read.

NullPointerException if buffer is Null.

IndexOutOfBoundsException if offset + length is greater than buffer size.

6.801.3.17 `virtual void decaf::internal::net::tcp::TcpSocket::setOption (int option,
int value) throw (decaf::io::IOException) [virtual]`

Sets the specified option on the **Socket** (p.3281) if supported.

Parameters

option The **Socket** (p.3281) option to set.

value The value of the socket option to apply to the socket.

Exceptions

IOException if an I/O error occurs while performing this operation.

Implements **decaf::net::SocketImpl** (p.3312).

6.801.3.18 `virtual void decaf::internal::net::tcp::TcpSocket::shutdownInput ()
throw (decaf::io::IOException) [virtual]`

Places the input stream for this socket at "end of stream".

Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p.3312) on the socket, the stream will return EOF.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p.3312).

6.801.3.19 `virtual void decaf::internal::net::tcp::TcpSocket::shutdownOutput ()
throw (decaf::io::IOException) [virtual]`

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p.3313) on the socket, the stream will throw an *IOException*.

Exceptions

IOException if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p.3313).

6.801.3.20 `void decaf::internal::net::tcp::TcpSocket::write (
const unsigned char * buffer, int size, int
offset, int length) throw (decaf::io::IOException,
decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::NullPointerException)`

Writes the specified data in the passed in buffer to the **Socket**.

Parameters

- buffer* The buffer to write to the socket.
- size* The size of the specified buffer.
- offset* The offset into the buffer where the data to write starts at.
- length* The number of bytes past offset to write.

Exceptions

- IOException* if an I/O error occurs during the write.
- NullPointerException* if buffer is Null.
- IndexOutOfBoundsException* if offset + length is greater than buffer size.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/**TcpSocket.h**

6.802 decaf::internal::net::tcp::TcpSocketInputStream Class Reference

Input stream for performing reads on a socket.

```
#include <src/main/decaf/internal/net/tcp/TcpSocketInputStream.h>
```

Inheritance diagram for decaf::internal::net::tcp::TcpSocketInputStream:

Public Member Functions

- **TcpSocketInputStream** (**TcpSocket** *socket)
Create a new InputStream to use for reading from the TCP/IP socket.
- virtual **~TcpSocketInputStream** ()
- virtual int **available** () const throw (decaf::io::IOException)
*Indicates the number of bytes available.
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.
The default implementation of this method returns zero.*
Returns
the number of bytes available on this input stream.
Exceptions
IOException (p. 2003) if an I/O error occurs.
- virtual void **close** () throw (decaf::io::IOException)
Close - does nothing.
- virtual long long **skip** (long long num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Not supported.

Protected Member Functions

- virtual int **doReadByte** () throw (io::IOException)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

6.802.1 Detailed Description

Input stream for performing reads on a socket. This class will only work properly for blocking sockets.

Since

1.0

6.802.2 Constructor & Destructor Documentation

6.802.2.1 decaf::internal::net::tcp::TcpSocketInputStream::TcpSocketInputStream (TcpSocket * *socket*)

Create a new InputStream to use for reading from the TCP/IP socket.

Parameters

socket The parent SocketImpl for this stream.

6.802.2.2 virtual decaf::internal::net::tcp::TcpSocketInputStream::~~TcpSocketInputStream () [virtual]

6.802.3 Member Function Documentation

6.802.3.1 virtual int decaf::internal::net::tcp::TcpSocketInputStream::available () const throw (decaf::io::IOException) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 2003) if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1911).

6.802.3.2 virtual void decaf::internal::net::tcp::TcpSocketInputStream::close ()
throw (decaf::io::IOException) [virtual]

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Closes the **InputStream** (p. 1909) freeing any resources that might have been aquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::InputStream** (p. 1912).

6.802.3.3 virtual int de-
caf::internal::net::tcp::TcpSocketInputStream::doReadArrayBounded
(unsigned char * *buffer*, int *size*, int *off-*
set, int *length*) throw (decaf::io::IOException,
decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::NullPointerException) [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1912).

6.802.3.4 virtual int decaf::internal::net::tcp::TcpSocketInputStream::doReadByte ()
throw (io::IOException) [protected, virtual]

Implements **decaf::io::InputStream** (p. 1912).

6.802.3.5 virtual long long decaf::internal::net::tcp::TcpSocketInputStream::skip
(long long *num*) throw (decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Not supported.

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1909) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num The number of bytes to skip.

Returns

total bytes skipped

Exceptions

IOException (p. 2003) if an I/O error occurs.

UnsupportedOperationException if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1917).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/tcp/TcpSocketInputStream.h`

6.803 decaf::internal::net::tcp::TcpSocketOutputStream Class Reference

Output stream for performing write operations on a socket.

```
#include <src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h>
```

Inheritance diagram for `decaf::internal::net::tcp::TcpSocketOutputStream`:

Public Member Functions

- **TcpSocketOutputStream** (**TcpSocket** *socket)
Create a new instance of a Socket OutputStream class.
- virtual **~TcpSocketOutputStream** ()
- virtual void **close** () throw (decaf::io::IOException)
The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char c) throw (decaf::io::IOException)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

6.803.1 Detailed Description

Output stream for performing write operations on a socket.

Since

1.0

6.803.2 Constructor & Destructor Documentation

6.803.2.1 decaf::internal::net::tcp::TcpSocketOutputStream::TcpSocketOutputStream (**TcpSocket** * *socket*)

Create a new instance of a Socket OutputStream class.

Parameters

socket The socket to use to write out the data.

6.803.2.2 virtual
 decaf::internal::net::tcp::TcpSocketOutputStream::~~TcpSocketOutputStream
 () [virtual]

6.803.3 Member Function Documentation

6.803.3.1 virtual void decaf::internal::net::tcp::TcpSocketOutputStream::close ()
 throw (decaf::io::IOException) [virtual]

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2720).

6.803.3.2 virtual void de-
 caf::internal::net::tcp::TcpSocketOutputStream::doWriteArrayBounded (
 const unsigned char * *buffer*, int *size*, int *offset*, int *length*) throw
 (decaf::io::IOException, decaf::lang::exceptions::NullPointerException,
 decaf::lang::exceptions::IndexOutOfBoundsException) [protected,
 virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2721).

6.803.3.3 virtual void de-
 caf::internal::net::tcp::TcpSocketOutputStream::doWriteByte (unsigned
 char *c*) throw (decaf::io::IOException) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2721).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h

6.804 activemq::transport::tcp::TcpTransport Class Reference

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2005).

```
#include <src/main/activemq/transport/tcp/TcpTransport.h>
```

Inheritance diagram for activemq::transport::tcp::TcpTransport:

Public Member Functions

- **TcpTransport** (const **Pointer**< **Transport** > &next)

*Creates a new instance of a **TcpTransport** (p. 3510), the transport is left unconnected and is in a unusable state until the connect method is called.*

- virtual `~TcpTransport ()`
- virtual `void connect (const decaf::net::URI &uri, const decaf::util::Properties &properties)`
Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in.
- virtual `void close () throw (decaf::io::IOException)`
Delegates to the superclass and then closes the socket.
- virtual `bool isFaultTolerant () const`
*Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual `bool isConnected () const`
*Is the **Transport** (p. 3629) Connected to its Broker.*
- virtual `bool isClosed () const`
*Has the **Transport** (p. 3629) been shutdown and no longer usable.*

Protected Member Functions

- virtual `decaf::net::Socket * createSocket ()`
Create an unconnected Socket instance to be used by the transport to communicate with the broker.
- virtual `void configureSocket (decaf::net::Socket *socket, const decaf::util::Properties &properties)`
Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server.

6.804.1 Detailed Description

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2005). The lower level transport should take care of managing stream reads and writes.

6.804.2 Constructor & Destructor Documentation

6.804.2.1 `activemq::transport::tcp::TcpTransport::TcpTransport (const Pointer< Transport > & next)`

Creates a new instance of a **TcpTransport** (p. 3510), the transport is left unconnected and is in a unusable state until the connect method is called.

Parameters

next The next transport in the chain

6.804.2.2 virtual `activemq::transport::tcp::TcpTransport::~~TcpTransport ()`
[virtual]

6.804.3 Member Function Documentation

6.804.3.1 virtual void `activemq::transport::tcp::TcpTransport::close ()` throw (`decaf::io::IOException`) [virtual]

Delegates to the superclass and then closes the socket.

Exceptions

IOException if errors occur.

Reimplemented from `activemq::transport::TransportFilter` (p. 3638).

6.804.3.2 virtual void `activemq::transport::tcp::TcpTransport::configureSocket (decaf::net::Socket * socket, const decaf::util::Properties & properties)`
[protected, virtual]

Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server.

Subclasses can override this option to set more configuration options, they should called the base class version to allow the default set of Socket options to also be configured.

Parameters

socket The Socket instance to configure using options from the given Properties.

Exceptions

NullPointerException if the Socket instance is null.

IllegalArgumentException if the socket instance is not handled by the class.

SocketException if there is an error while setting one of the Socket options.

6.804.3.3 void `activemq::transport::tcp::TcpTransport::connect (const decaf::net::URI & uri, const decaf::util::Properties & properties)`

Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in.

The Socket is configured using parameters in the properties that are passed to this method.

Parameters

uri The URI that the **Transport** (p. 3629) is to connect to once initialized.

properties The Properties that have been parsed from the URI or from configuration files.

6.804.3.4 `virtual decaf::net::Socket* activemq::transport::tcp::TcpTransport::createSocket ()`
[protected, virtual]

Create an unconnected Socket instance to be used by the transport to communicate with the broker.

Returns

a newly created unconnected Socket instance.

Exceptions

IOException if there is an error while creating the unconnected Socket.

Reimplemented in `activemq::transport::tcp::SslTransport` (p. 3349).

6.804.3.5 `virtual bool activemq::transport::tcp::TcpTransport::isClosed () const`
[inline, virtual]

Has the **Transport** (p. 3629) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3629)

Reimplemented from `activemq::transport::TransportFilter` (p. 3639).

6.804.3.6 `virtual bool activemq::transport::tcp::TcpTransport::isConnected () const`
[inline, virtual]

Is the **Transport** (p. 3629) Connected to its Broker.

Returns

true if a connection has been made.

Reimplemented from `activemq::transport::TransportFilter` (p. 3639).

6.804.3.7 `virtual bool activemq::transport::tcp::TcpTransport::isFaultTolerant () const`
[inline, virtual]

Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3629) is fault tolerant.

Reimplemented from `activemq::transport::TransportFilter` (p. 3640).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/TcpTransport.h`

6.805 activemq::transport::tcp::TcpTransportFactory Class Reference

Factory Responsible for creating the **TcpTransport** (p. 3510).

```
#include <src/main/activemq/transport/tcp/TcpTransportFactory.h>
```

Inheritance diagram for activemq::transport::tcp::TcpTransportFactory:

Public Member Functions

- virtual **~TcpTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

*Creates a fully configured **Transport** (p. 3629) instance which could be a chain of filters and transports.*

- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

*Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **Pointer< wireformat::WireFormat >** &wireFormat, const **decaf::util::Properties** &properties) throw (exceptions::ActiveMQException)

*Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.*

6.805.1 Detailed Description

Factory Responsible for creating the **TcpTransport** (p. 3510).

6.805.2 Constructor & Destructor Documentation

6.805.2.1 `virtual`
`activemq::transport::tcp::TcpTransportFactory::~~TcpTransportFactory (`
`) [inline, virtual]`

6.805.3 Member Function Documentation

6.805.3.1 `virtual Pointer<Transport> ac-`
`tivemq::transport::tcp::TcpTransportFactory::create`
`(const decaf::net::URI & location) throw (`
`exceptions::ActiveMQException) [virtual]`

Creates a fully configured **Transport** (p. 3629) instance which could be a chain of filters and transports.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements `activemq::transport::TransportFactory` (p. 3635).

6.805.3.2 `virtual Pointer<Transport> ac-`
`tivemq::transport::tcp::TcpTransportFactory::createComposite (const`
`decaf::net::URI & location) throw (exceptions::ActiveMQException)`
`[virtual]`

Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements `activemq::transport::TransportFactory` (p. 3635).

6.805.3.3 `virtual Pointer<Transport> ac-`
`tivemq::transport::tcp::TcpTransportFactory::doCreateComposite (const`
`decaf::net::URI & location, const Pointer< wireformat::WireFormat >`
`& wireFormat, const decaf::util::Properties & properties) throw (`
`exceptions::ActiveMQException) [protected, virtual]`

Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.

Parameters

- location* - URI location to connect to.
- wireFormat* - the assigned WireFormat for the new **Transport** (p. 3629).
- properties* - Properties to apply to the transport.

Returns

new Pointer to a **TcpTransport** (p. 3510).

Exceptions

ActiveMQException if an error occurs

Reimplemented in **activemq::transport::tcp::SslTransportFactory** (p. 3350).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/TcpTransportFactory.h

6.806 cms::TemporaryQueue Class Reference

Defines a Temporary **Queue** (p. 2947) based **Destination** (p. 1610).

```
#include <src/main/cms/TemporaryQueue.h>
```

Inheritance diagram for cms::TemporaryQueue:

Public Member Functions

- virtual **~TemporaryQueue** ()
- virtual std::string **getQueueName** () const =0 throw (CMSEException)
Gets the name of this queue.
- virtual void **destroy** ()=0 throw (CMSEException)
*Destroy's the Temporary **Destination** (p. 1610) at the Provider.*

6.806.1 Detailed Description

Defines a Temporary **Queue** (p. 2947) based **Destination** (p. 1610). A **TemporaryQueue** (p. 3516) is a special type of **Queue** (p. 2947) **Destination** (p. 1610) that can only be consumed from the **Connection** (p. 1168) which created it. TemporaryQueues are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryQueue** (p. 3516) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 1168) that created it.

Since

1.0

6.806.2 Constructor & Destructor Documentation

6.806.2.1 `virtual cms::TemporaryQueue::~TemporaryQueue () [inline, virtual]`

6.806.3 Member Function Documentation

6.806.3.1 `virtual void cms::TemporaryQueue::destroy () throw (CMSEException) [pure virtual]`

Destroy's the Temporary **Destination** (p. 1610) at the Provider.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQTempQueue` (p. 551).

6.806.3.2 `virtual std::string cms::TemporaryQueue::getQueueName () const throw (CMSEException) [pure virtual]`

Gets the name of this queue.

Returns

The queue name.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQTempQueue` (p. 552).

The documentation for this class was generated from the following file:

- `src/main/cms/TemporaryQueue.h`

6.807 cms::TemporaryTopic Class Reference

Defines a Temporary **Topic** (p. 3568) based **Destination** (p. 1610).

```
#include <src/main/cms/TemporaryTopic.h>
```

Inheritance diagram for cms::TemporaryTopic:

Public Member Functions

- `virtual ~TemporaryTopic ()`
- `virtual std::string getTopicName () const =0 throw (CMSEException)`
Gets the name of this topic.

- virtual void **destroy** ()=0 throw (CMSEException)
Destroy's the Temporary Destination (p. 1610) at the Provider.

6.807.1 Detailed Description

Defines a Temporary **Topic** (p. 3568) based **Destination** (p. 1610). A **TemporaryTopic** (p. 3517) is a special type of **Topic** (p. 3568) **Destination** (p. 1610) that can only be consumed from the **Connection** (p. 1168) which created it. TemporaryTopics are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryTopic** (p. 3517) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 1168) that created it.

Since

1.0

6.807.2 Constructor & Destructor Documentation

6.807.2.1 virtual cms::TemporaryTopic::~TemporaryTopic () [inline, virtual]

6.807.3 Member Function Documentation

6.807.3.1 virtual void cms::TemporaryTopic::destroy () throw (CMSEException) [pure virtual]

Destroy's the Temporary **Destination** (p. 1610) at the Provider.

Exceptions

CMSEException (p. 1074)

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 579).

6.807.3.2 virtual std::string cms::TemporaryTopic::getTopicName () const throw (CMSEException) [pure virtual]

Gets the name of this topic.

Returns

The topic name.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 580).

The documentation for this class was generated from the following file:

- src/main/cms/**TemporaryTopic.h**

6.808 cms::TextMessage Class Reference

Interface for a text message.

```
#include <src/main/cms/TextMessage.h>
```

Inheritance diagram for cms::TextMessage:

Public Member Functions

- virtual `~TextMessage` ()
- virtual `std::string getText` () const =0 throw (cms::CMSEException)
Gets the message character buffer.
- virtual void `setText` (const char *msg)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets the message contents, does not take ownership of the passed char, but copies it instead.*
- virtual void `setText` (const std::string &msg)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets the message contents.

6.808.1 Detailed Description

Interface for a text message. A **TextMessage** (p. 3519) can contain any Text based pay load such as an XML Document or other Text based document.

Like all Messages, a **TextMessage** (p. 3519) is received in Read-Only mode, any attempt to write to the **Message** (p. 2375) will result in a MessageNotWritableException being thrown until the `clearBody` method is called which will erase the contents and place the message back in a read / write mode.

Since

1.0

6.808.2 Constructor & Destructor Documentation

6.808.2.1 virtual cms::TextMessage::~TextMessage () [inline, virtual]

6.808.3 Member Function Documentation

6.808.3.1 virtual std::string cms::TextMessage::getText () const throw (cms::CMSEException) [pure virtual]

Gets the message character buffer.

Returns

The message character buffer.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

```
6.808.3.2 virtual void cms::TextMessage::setText ( const std::string & msg )
          throw ( cms::MessageNotWriteableException, cms::CMSEException )
          [pure virtual]
```

Sets the message contents.

Parameters

msg The message buffer.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode..

```
6.808.3.3 virtual void cms::TextMessage::setText ( const char * msg ) throw
          ( cms::MessageNotWriteableException, cms::CMSEException ) [pure
          virtual]
```

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters

msg The message buffer.

Exceptions

CMSEException (p. 1074) - if an internal error occurs.

MessageNotWriteableException (p. 2549) - if the message is in read-only mode..

The documentation for this class was generated from the following file:

- src/main/cms/TextMessage.h

6.809 decaf::lang::Thread Class Reference

A **Thread** (p. 3520) is a concurrent unit of execution.

```
#include <src/main/decaf/lang/Thread.h>
```

Inheritance diagram for decaf::lang::Thread:

Data Structures

- class **UncaughtExceptionHandler**

Interface for handlers invoked when a Thread (p. 3520) abruptly terminates due to an uncaught exception.

Public Types

- enum **State** {
NEW = 0, **RUNNABLE** = 1, **BLOCKED** = 2, **WAITING** = 3,
TIMED_WAITING = 4, **SLEEPING** = 5, **TERMINATED** = 6 }
*Represents the various states that the **Thread** (p. 3520) can be in during its lifetime.*

Public Member Functions

- **Thread** ()
*Constructs a new **Thread** (p. 3520).*
- **Thread** (**Runnable** *task)
*Constructs a new **Thread** (p. 3520) with the given target **Runnable** (p. 3111) task.*
- **Thread** (const std::string &name)
*Constructs a new **Thread** (p. 3520) with the given name.*
- **Thread** (**Runnable** *task, const std::string &name)
*Constructs a new **Thread** (p. 3520) with the given target **Runnable** (p. 3111) task and name.*
- virtual ~**Thread** ()
- virtual void **start** () throw (decaf::lang::exceptions::IllegalThreadStateException, decaf::lang::exceptions::RuntimeException)
Creates a system thread and starts it in a joinable mode.
- virtual void **join** () throw (decaf::lang::exceptions::InterruptedException)
*Forces the Current **Thread** (p. 3520) to wait until the thread exits.*
- virtual void **join** (long long millisecs) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException)
*Forces the Current **Thread** (p. 3520) to wait until the thread exits.*
- virtual void **join** (long long millisecs, unsigned int nanos) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException)
*Forces the Current **Thread** (p. 3520) to wait until the thread exits.*
- virtual void **run** ()
Default implementation of the run method - does nothing.
- std::string **getName** () const
Returns the Thread's assigned name.
- void **setName** (const std::string &name)
*Sets the name of the **Thread** (p. 3520) to the new Name given by the argument name*
- int **getPriority** () const

*Gets the currently set priority for this **Thread** (p. 3520).*

- void **setPriority** (int value) throw (decaf::lang::exceptions::IllegalArgumentException)
Sets the current Thread's priority to the newly specified value.
- const **UncaughtExceptionHandler** * **getUncaughtExceptionHandler** () const
Set the handler invoked when this thread abruptly terminates due to an uncaught exception.
- void **setUncaughtExceptionHandler** (**UncaughtExceptionHandler** *handler)
Set the handler invoked when this thread abruptly terminates due to an uncaught exception.
- std::string **toString** () const
*Returns a string that describes the **Thread** (p. 3520).*
- bool **isAlive** () const
*Returns true if the **Thread** (p. 3520) is alive, meaning it has been started and has not yet died.*
- **Thread::State** **getState** () const
*Returns the currently set State of this **Thread** (p. 3520).*

Static Public Member Functions

- static void **sleep** (long long millisecs) throw (lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException)
Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.
- static void **sleep** (long long millisecs, unsigned int nanos) throw (lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException)
Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers.
- static void **yield** ()
Causes the currently executing thread object to temporarily pause and allow other threads to execute.
- static long long **getId** ()
*Obtains the **Thread** (p. 3520) Id of the current thread.*
- static **Thread** * **currentThread** ()
Returns a pointer to the currently executing thread object.

Static Public Attributes

- static const int **MIN_PRIORITY** = 1
The minimum priority that a thread can have.

- static const int **NORM_PRIORITY** = 5

The default priority that a thread is given at create time.

- static const int **MAX_PRIORITY** = 10

The maximum priority that a thread can have.

Friends

- class **decaf::util::concurrent::locks::LockSupport**
- class **decaf::lang::Runtime**

6.809.1 Detailed Description

A **Thread** (p. 3520) is a concurrent unit of execution. It has its own call stack for methods being invoked, their arguments and local variables. Each process has at least one main **Thread** (p. 3520) running when it is started; typically, there are several others for housekeeping. The application might decide to launch additional Threads for specific purposes.

Threads in the same process interact and synchronize by the use of shared objects and monitors associated with these objects.

There are basically two main ways of having a **Thread** (p. 3520) execute application code. One is providing a new class that extends **Thread** (p. 3520) and overriding its **run()** (p. 3527) method. The other is providing a new **Thread** (p. 3520) instance with a **Runnable** (p. 3111) object during its creation. In both cases, the **start()** (p. 3528) method must be called to actually execute the new **Thread** (p. 3520).

Each **Thread** (p. 3520) has an integer priority that basically determines the amount of CPU time the **Thread** (p. 3520) gets. It can be set using the **setPriority(int)** (p. 3527) method. A **Thread** (p. 3520) can also be made a daemon, which makes it run in the background. The latter also affects VM termination behavior: the VM does not terminate automatically as long as there are non-daemon threads running.

See also

decaf.lang.ThreadGroup (p. 3531)

Since

1.0

6.809.2 Member Enumeration Documentation

6.809.2.1 enum **decaf::lang::Thread::State**

Represents the various states that the **Thread** (p. 3520) can be in during its lifetime.

Enumerator:

NEW Before a **Thread** (p. 3520) is started it exists in this State.

RUNNABLE While a **Thread** (p. 3520) is running and is not blocked it is in this State.

BLOCKED A **Thread** (p. 3520) that is waiting to acquire a lock is in this state.

WAITING A **Thread** (p. 3520) that is waiting for another **Thread** (p. 3520) to perform an action is in this state.

TIMED_WAITING A **Thread** (p. 3520) that is waiting for another **Thread** (p. 3520) to perform an action up to a specified time interval is in this state.

SLEEPING A **Thread** (p. 3520) that is blocked in a **Sleep** call is in this state.

TERMINATED A **Thread** (p. 3520) whose **run** method has exited is in this state.

6.809.3 Constructor & Destructor Documentation

6.809.3.1 decaf::lang::Thread::Thread ()

Constructs a new **Thread** (p. 3520).

This constructor has the same effect as **Thread**(**NULL**, **NULL**, **GIVEN_NAME**), where **GIVEN_NAME** is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

6.809.3.2 decaf::lang::Thread::Thread (**Runnable** * *task*)

Constructs a new **Thread** (p. 3520) with the given target **Runnable** (p. 3111) task.

This constructor has the same effect as **Thread**(**NULL**, *task*, **GIVEN_NAME**), where **GIVEN_NAME** is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

Parameters

task the **Runnable** (p. 3111) that this thread manages, if the task is **NULL** the **Thread**'s **run** method is used instead.

6.809.3.3 decaf::lang::Thread::Thread (**const** **std::string** & *name*)

Constructs a new **Thread** (p. 3520) with the given name.

This constructor has the same effect as **Thread**(**NULL**, **NULL**, **GIVEN_NAME**), where **GIVEN_NAME** is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

Parameters

name the name to assign to this **Thread** (p. 3520).

6.809.3.4 decaf::lang::Thread::Thread (**Runnable** * *task*, **const** **std::string** & *name*)

Constructs a new **Thread** (p. 3520) with the given target **Runnable** (p. 3111) task and name.

This constructor has the same effect as **Thread**(**NULL**, *task*, **GIVEN_NAME**), where **GIVEN_NAME** is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

Parameters

task the **Runnable** (p. 3111) that this thread manages, if the task is NULL the Thread's run method is used instead.

name the name to assign to this **Thread** (p. 3520).

6.809.3.5 `virtual decaf::lang::Thread::~~Thread () [virtual]`

6.809.4 Member Function Documentation

6.809.4.1 `static Thread* decaf::lang::Thread::currentThread () [static]`

Returns a pointer to the currently executing thread object.

Returns

Pointer (p. 2756) to the **Thread** (p. 3520) object representing the currently running **Thread** (p. 3520).

6.809.4.2 `static long long decaf::lang::Thread::getId () [static]`

Obtains the **Thread** (p. 3520) Id of the current thread.

Returns

Thread (p. 3520) Id

6.809.4.3 `std::string decaf::lang::Thread::getName () const`

Returns the Thread's assigned name.

Returns

the Name of the **Thread** (p. 3520).

6.809.4.4 `int decaf::lang::Thread::getPriority () const`

Gets the currently set priority for this **Thread** (p. 3520).

Returns

an int value representing the Thread's current priority.

6.809.4.5 `Thread::State decaf::lang::Thread::getState () const`

Returns the currently set State of this **Thread** (p. 3520).

Returns

the Thread's current state.

6.809.4.6 `const UncaughtExceptionHandler* decaf::lang::Thread::getUncaughtExceptionHandler () const`

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

Returns

a pointer to the set **UncaughtExceptionHandler** (p. 3648).

6.809.4.7 `bool decaf::lang::Thread::isAlive () const`

Returns true if the **Thread** (p. 3520) is alive, meaning it has been started and has not yet died.

Returns

true if the thread is alive.

6.809.4.8 `virtual void decaf::lang::Thread::join (long long millisecs, unsigned int nanos) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException) [virtual]`

Forces the Current **Thread** (p. 3520) to wait until the thread exits.

Parameters

millisecs the time in Milliseconds before the thread resumes

nanos 0-999999 extra nanoseconds to sleep.

Exceptions

IllegalArgumentException if the nanoseconds parameter is out of range or the milliseconds paramter is negative.

InterruptedException if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

6.809.4.9 `virtual void decaf::lang::Thread::join () throw (decaf::lang::exceptions::InterruptedException) [virtual]`

Forces the Current **Thread** (p. 3520) to wait until the thread exits.

Exceptions

InterruptedException if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

6.809.4.10 `virtual void decaf::lang::Thread::join (long long millisecs)
 throw (decaf::lang::exceptions::IllegalArgumentException,
 decaf::lang::exceptions::InterruptedException) [virtual]`

Forces the Current **Thread** (p. 3520) to wait until the thread exits.

Parameters

millisecs the time in Milliseconds before the thread resumes

Exceptions

IllegalArgumentException if the milliseconds parameter is negative.

InterruptedException if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

6.809.4.11 `virtual void decaf::lang::Thread::run () [virtual]`

Default implementation of the run method - does nothing.

Implements **decaf::lang::Runnable** (p. 3112).

Reimplemented in **activemq::transport::mock::InternalCommandListener** (p. 1987), and **decaf::util::concurrent::PooledThread** (p. 2778).

6.809.4.12 `void decaf::lang::Thread::setName (const std::string & name)`

Sets the name of the **Thread** (p. 3520) to the new Name given by the argument *name*
 name the new name of the **Thread** (p. 3520).

6.809.4.13 `void decaf::lang::Thread::setPriority (int value) throw (decaf::lang::exceptions::IllegalArgumentException)`

Sets the current Thread's priority to the newly specified value.

The given value must be within the range **Thread::MIN_PRIORITY** (p. 3529) and **Thread::MAX_PRIORITY** (p. 3529).

Parameters

value the new priority value to assign to this **Thread** (p. 3520).

Exceptions

IllegalArgumentException if the value is out of range.

6.809.4.14 `void decaf::lang::Thread::setUncaughtExceptionHandler (UncaughtExceptionHandler * handler)`

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

Parameters

handler the `UncaughtExceptionHandler` to invoke when the **Thread** (p. 3520) terminates due to an uncaught exception.

6.809.4.15 `static void decaf::lang::Thread::sleep (long long millisecs, unsigned int nanos) throw (lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException) [static]`

Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers.

Note that this method is a static method that applies to the calling thread and not to the thread object.

Parameters

millisecs time in milliseconds to halt execution.

nanos 0-999999 extra nanoseconds to sleep.

Exceptions

IllegalArgumentException if the nanoseconds parameter is out of range or the milliseconds parameter is negative.

InterruptedException if the **Thread** (p. 3520) was interrupted while sleeping.

6.809.4.16 `static void decaf::lang::Thread::sleep (long long millisecs) throw (lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException) [static]`

Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.

Note that this method is a static method that applies to the calling thread and not to the thread object.

Parameters

millisecs time in milliseconds to halt execution.

Exceptions

IllegalArgumentException if the milliseconds parameter is negative.

InterruptedException if the **Thread** (p. 3520) was interrupted while sleeping.

6.809.4.17 `virtual void decaf::lang::Thread::start () throw (decaf::lang::exceptions::IllegalThreadStateException, decaf::lang::exceptions::RuntimeException) [virtual]`

Creates a system thread and starts it in a joinable mode.

Upon creation, the `run()` (p. 3527) method of either this object or the provided **Runnable** (p. 3111) object will be invoked in the context of this thread.

Exceptions

IllegalThreadStateException if the thread has already been started.

RuntimeException if the **Thread** (p. 3520) cannot be created for some reason.

6.809.4.18 `std::string decaf::lang::Thread::toString () const`

Returns a string that describes the **Thread** (p. 3520).

Returns

string describing the **Thread** (p. 3520).

6.809.4.19 `static void decaf::lang::Thread::yield () [static]`

Causes the currently executing thread object to temporarily pause and allow other threads to execute.

6.809.5 Friends And Related Function Documentation

6.809.5.1 `friend class decaf::lang::Runtime [friend]`

6.809.5.2 `friend class decaf::util::concurrent::locks::LockSupport [friend]`

6.809.6 Field Documentation

6.809.6.1 `const int decaf::lang::Thread::MAX_PRIORITY = 10 [static]`

The maximum priority that a thread can have.

6.809.6.2 `const int decaf::lang::Thread::MIN_PRIORITY = 1 [static]`

The minimum priority that a thread can have.

6.809.6.3 `const int decaf::lang::Thread::NORM_PRIORITY = 5 [static]`

The default priority that a thread is given at create time.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Thread.h`

6.810 `decaf::util::concurrent::ThreadFactory` Class Reference

public interface **ThreadFactory** (p. 3529)

```
#include <src/main/decaf/util/concurrent/ThreadFactory.h>
```

Public Member Functions

- virtual `~ThreadFactory()`
- virtual `decaf::lang::Thread * newThread (decaf::lang::Runnable *r)=0`
Constructs a new Thread.

6.810.1 Detailed Description

public interface **ThreadFactory** (p. 3529) An object that creates new threads on demand. Using thread factories removes hardwiring of calls to new Thread, enabling applications to use special thread subclasses, priorities, etc.

The simplest implementation of this interface is just:

```
class SimpleThreadFactory : public ThreadFactory (p. 3529) { public: Thread* newThread(
Runnable* r ) { return new Thread(r); } }
```

The Executors.defaultThreadFactory() method provides a more useful simple implementation, that sets the created thread context to known values before returning it.

Since

1.0

6.810.2 Constructor & Destructor Documentation

- 6.810.2.1** virtual `decaf::util::concurrent::ThreadFactory::~~ThreadFactory ()`
[inline, virtual]

6.810.3 Member Function Documentation

- 6.810.3.1** virtual `decaf::lang::Thread* decaf::util::concurrent::ThreadFactory::newThread (decaf::lang::Runnable * r)` [pure virtual]

Constructs a new Thread.

Implementations may also initialize priority, name, daemon status, ThreadGroup, etc. The pointer passed is still owned by the caller and is not deleted by the Thread object. The caller owns the returned Thread object and must delete it when finished.

Parameters

r A pointer to a Runnable instance to be executed by new Thread instance returned.

Returns

constructed thread, or NULL if the request to create a thread is rejected the caller owns the returned pointer.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadFactory.h**

6.811 decaf::lang::ThreadGroup Class Reference

```
#include <src/main/decaf/lang/ThreadGroup.h>
```

Public Member Functions

- **ThreadGroup** ()
- virtual **~ThreadGroup** ()

6.811.1 Detailed Description

Since

1.0

6.811.2 Constructor & Destructor Documentation

6.811.2.1 decaf::lang::ThreadGroup::ThreadGroup ()

6.811.2.2 virtual decaf::lang::ThreadGroup::~~ThreadGroup () [virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/ThreadGroup.h`

6.812 decaf::util::concurrent::ThreadPool Class Reference

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

```
#include <src/main/decaf/util/concurrent/ThreadPool.h>
```

Inheritance diagram for decaf::util::concurrent::ThreadPool:

Public Types

- typedef std::pair< lang::Runnable *, TaskListener * > **Task**

Public Member Functions

- **ThreadPool** ()
- virtual **~ThreadPool** ()
- virtual void **queueTask** (**Task** task) throw (lang::Exception)
Queue (p. 2948) a task to be completed by one of the Pooled Threads.
- virtual **Task deQueueTask** () throw (lang::Exception)
DeQueue a task to be completed by one of the Pooled Threads.

- virtual `std::size_t` **getPoolSize** () const
Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.
- virtual `std::size_t` **getBacklog** () const
Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.
- virtual void **reserve** (std::size_t size)
Ensures that there is at least the specified number of Threads allocated to the pool.
- virtual `std::size_t` **getMaxThreads** () const
Get the Max Number of Threads this Pool can contain.
- virtual void **setMaxThreads** (std::size_t maxThreads)
Sets the Max number of threads this pool can contain.
- virtual `std::size_t` **getBlockSize** () const
Gets the Max number of threads that can be allocated at a time when new threads are needed.
- virtual void **setBlockSize** (std::size_t blockSize)
Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.
- virtual `std::size_t` **getFreeThreadCount** () const
Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable.
- virtual void **onTaskStarted** (PooledThread *thread)
Called by a pooled thread when it is about to begin executing a new task.
- virtual void **onTaskCompleted** (PooledThread *thread)
Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.
- virtual void **onTaskException** (PooledThread *thread, lang::Exception &ex)
*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2777) is now no longer running.*

Static Public Member Functions

- static `ThreadPool *` **getInstance** ()
Return the one and only Thread Pool instance.

Static Public Attributes

- static const `size_t` **DEFAULT_MAX_POOL_SIZE** = 10
- static const `size_t` **DEFAULT_MAX_BLOCK_SIZE** = 3

6.812.1 Detailed Description

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks. The Thread Pool has max size that it will grow to. The thread pool allocates threads in blocks. When there are no waiting worker threads and a task is queued then a new batch is allocated. The user can specify the size of the blocks, otherwise a default value is used.

When the user queues a task they must also queue a listener to be notified when the task has completed, this provides the user with a mechanism to know when a task object can be freed.

To have the Thread Pool perform a task, the user enqueue's an object that implements the `Runnable` interface and one of the worker threads will be executing it in its thread context.

6.812.2 Member Typedef Documentation

6.812.2.1 `typedef std::pair<lang::Runnable*, TaskListener*>
decaf::util::concurrent::ThreadPool::Task`

6.812.3 Constructor & Destructor Documentation

6.812.3.1 `decaf::util::concurrent::ThreadPool::ThreadPool ()`

6.812.3.2 `virtual decaf::util::concurrent::ThreadPool::~~ThreadPool () [virtual]`

6.812.4 Member Function Documentation

6.812.4.1 `virtual Task decaf::util::concurrent::ThreadPool::deQueueTask ()
throw (lang::Exception) [virtual]`

DeQueue a task to be completed by one of the Pooled Threads.

A caller of this method will block until there is something in the tasks queue, therefore care must be taken when calling this function. Normally clients of **ThreadPool** (p. 3531) don't use this, only the **PooledThread** (p. 2777) objects owned by this **ThreadPool** (p. 3531).

Returns

object that derives from `Runnable`

Exceptions

ActiveMQException

6.812.4.2 `virtual std::size_t decaf::util::concurrent::ThreadPool::getBacklog ()
const [inline, virtual]`

Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.

Returns

number of outstanding tasks.

6.812.4.3 `virtual std::size_t decaf::util::concurrent::ThreadPool::getBlockSize () const [inline, virtual]`

Gets the Max number of threads that can be allocated at a time when new threads are needed.

Returns

max Thread Block Size

6.812.4.4 `virtual std::size_t decaf::util::concurrent::ThreadPool::getFreeThreadCount () const [inline, virtual]`

Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable.

This value could change immediately after calling as Threads could finish right after and be available again. This is informational only.

Returns

total free threads

6.812.4.5 `static ThreadPool* decaf::util::concurrent::ThreadPool::getInstance () [static]`

Return the one and only Thread Pool instance.

Returns

The Thread Pool Pointer

6.812.4.6 `virtual std::size_t decaf::util::concurrent::ThreadPool::getMaxThreads () const [inline, virtual]`

Get the Max Number of Threads this Pool can contain.

Returns

max size

6.812.4.7 `virtual std::size_t decaf::util::concurrent::ThreadPool::getPoolSize () const [inline, virtual]`

Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.

Returns

integer number of threads in existence.

6.812.4.8 **virtual void decaf::util::concurrent::ThreadPool::onTaskCompleted (PooledThread * *thread*) [virtual]**

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.

Parameters

thread Pointer the the Pooled Thread that is making this call.

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2780).

6.812.4.9 **virtual void decaf::util::concurrent::ThreadPool::onTaskException (PooledThread * *thread*, lang::Exception & *ex*) [virtual]**

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2777) is now no longer running.

Parameters

thread Pointer to the Pooled Thread that is making this call

ex The Exception that occurred.

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2780).

6.812.4.10 **virtual void decaf::util::concurrent::ThreadPool::onTaskStarted (PooledThread * *thread*) [virtual]**

Called by a pooled thread when it is about to begin executing a new task.

This will decrement the available threads counter so that this object knows when there are no more free threads and must create new ones.

Parameters

thread Pointer to the Pooled Thread that is making this call

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2780).

6.812.4.11 **virtual void decaf::util::concurrent::ThreadPool::queueTask (Task *task*) throw (lang::Exception) [virtual]**

Queue (p. 2948) a task to be completed by one of the Pooled Threads.

tasks are serviced as soon as a **PooledThread** (p. 2777) is available to run it.

Parameters

task object that derives from Runnable

Exceptions

ActiveMQException

6.812.4.12 virtual void decaf::util::concurrent::ThreadPool::reserve (std::size_t *size*) [virtual]

Ensures that there is at least the specified number of Threads allocated to the pool.

If the size is greater than the MAX number of threads in the pool, then only MAX threads are reserved. If the size is smaller than the number of threads currently in the pool, than nothing is done.

Parameters

size the number of threads to reserve.

6.812.4.13 virtual void decaf::util::concurrent::ThreadPool::setBlockSize (std::size_t *blockSize*) [virtual]

Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.

Parameters

blockSize Max Thread Block Size

6.812.4.14 virtual void decaf::util::concurrent::ThreadPool::setMaxThreads (std::size_t *maxThreads*) [virtual]

Sets the Max number of threads this pool can contain.

if this value is smaller than the current size of the pool nothing is done.

Parameters

maxThreads total number of threads that can be pooled

6.812.5 Field Documentation

6.812.5.1 const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_BLOCK_SIZE = 3 [static]

6.812.5.2 const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_POOL_SIZE = 10 [static]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/ThreadPool.h

6.813 decaf::lang::Throwable Class Reference

This class represents an error that has occurred.

```
#include <src/main/decaf/lang/Throwable.h>
```

Inheritance diagram for decaf::lang::Throwable:

Public Member Functions

- **Throwable** () throw ()
- virtual ~**Throwable** () throw ()
- virtual std::string **getMessage** () const =0
Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.
- virtual const std::exception * **getCause** () const =0
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- virtual void **initCause** (const std::exception *cause)=0
Initializes the contained cause exception with the one given.
- virtual void **setMark** (const char *file, const int lineNumber)=0
Adds a file/line number to the stack trace.
- virtual **Throwable** * **clone** () const =0
Clones this exception.
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const =0
Provides the stack trace for every point where this exception was caught, marked, and rethrown.
- virtual void **printStackTrace** () const =0
Prints the stack trace to std::err.
- virtual void **printStackTrace** (std::ostream &stream) const =0
Prints the stack trace to the given output stream.
- virtual std::string **getStackTraceString** () const =0
Gets the stack trace as one contiguous string.

6.813.1 Detailed Description

This class represents an error that has occurred. All Exceptions in the Decaf library should extend from this or from the **Exception** (p. 1712) class in order to ensure that all Decaf Exceptions are interchangeable with the std::exception class.

Throwable (p. 3536) can wrap another **Throwable** (p. 3536) as the cause if the error being thrown. The user can inspect the cause by calling **getCause**, the pointer returned is the property of the **Throwable** (p. 3536) instance and will be deleted when it is deleted or goes out of scope.

Since

1.0

6.813.2 Constructor & Destructor Documentation

6.813.2.1 decaf::lang::Throwable::Throwable () throw () [inline]

6.813.2.2 virtual decaf::lang::Throwable::~~Throwable () throw () [inline, virtual]

6.813.3 Member Function Documentation

6.813.3.1 virtual Throwable* decaf::lang::Throwable::clone () const [pure virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

Copy of this **Exception** (p.1712) object

Implemented in **activemq::exceptions::ActiveMQException** (p. 315),
activemq::exceptions::BrokerException (p. 798),
decaf::internal::net::ssl::openssl::OpenSSLSocketException (p. 2689),
decaf::io::EOFException (p. 1709),
decaf::io::InterruptedIOException (p. 1992),
decaf::io::IOException (p. 2005),
decaf::io::UnsupportedEncodingException (p. 3656),
decaf::io::UTFDataFormatException (p. 3705),
decaf::lang::Exception (p. 1715),
decaf::lang::exceptions::ClassCastException (p. 1064),
decaf::lang::exceptions::IllegalArgumentException (p. 1865),
decaf::lang::exceptions::IllegalMonitorStateException (p. 1867),
decaf::lang::exceptions::IllegalStateException (p. 1871),
decaf::lang::exceptions::IllegalThreadStateException (p. 1874),
decaf::lang::exceptions::IndexOutOfBoundsException (p. 1879),
decaf::lang::exceptions::InterruptedException (p. 1989),
decaf::lang::exceptions::InvalidStateException (p. 2002),
decaf::lang::exceptions::NoSuchElementException (p. 2647),
decaf::lang::exceptions::NullPointerException (p. 2652),
decaf::lang::exceptions::NumberFormatException (p. 2657),
decaf::lang::exceptions::RuntimeException (p. 3116),
decaf::lang::exceptions::UnsupportedOperationException (p. 3659),
decaf::net::BindException (p. 770),
decaf::net::ConnectException (p. 1167),
decaf::net::HttpRetryException (p. 1861),
decaf::net::MalformedURLException (p. 2305),
decaf::net::NoRouteToHostException (p. 2642),
decaf::net::PortUnreachableException (p. 2783),
decaf::net::ProtocolException (p. 2939),
decaf::net::SocketException (p. 3300),
decaf::net::SocketTimeoutException (p. 3321),
decaf::net::UnknownHostException (p. 3651),
decaf::net::UnknownServiceException (p. 3654),
decaf::net::URISyntaxException (p. 3689),
decaf::nio::BufferOverflowException (p. 882),
decaf::nio::BufferUnderflowException (p. 884),
decaf::nio::InvalidMarkException (p. 1999),
decaf::nio::ReadOnlyBufferException (p. 2968),
decaf::security::cert::CertificateEncodingException (p. 1011),
decaf::security::cert::CertificateException (p. 1013),
decaf::security::cert::CertificateExpiredException (p. 1015),
decaf::security::cert::CertificateNotYetValidException (p. 1017),
decaf::security::cert::CertificateParsingException (p. 1019),

decaf::security::GeneralSecurityException (p. 1847), **decaf::security::InvalidKeyException** (p. 1996), **decaf::security::KeyException** (p. 2153), **decaf::security::KeyManagementException** (p. 2155), **decaf::security::NoSuchAlgorithmException** (p. 2645), **decaf::security::NoSuchProviderException** (p. 2650), **decaf::security::SignatureException** (p. 3278), **decaf::util::concurrent::BrokenBarrierException** (p. 792), **decaf::util::concurrent::CancellationException** (p. 1006), **decaf::util::concurrent::ExecutionException** (p. 1749), **decaf::util::concurrent::RejectedExecutionException** (p. 2986), **decaf::util::concurrent::TimeoutException** (p. 3543), and **decaf::util::zip::DataFormatException** (p. 1452), and **decaf::util::zip::ZipException** (p. 3798).

6.813.3.2 `virtual const std::exception* decaf::lang::Throwable::getCause () const` [pure virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implemented in **decaf::lang::Exception** (p. 1716).

6.813.3.3 `virtual std::string decaf::lang::Throwable::getMessage () const` [pure virtual]

Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.

Returns

string errors message

Implemented in **decaf::lang::Exception** (p. 1716).

6.813.3.4 `virtual std::vector< std::pair< std::string, int> > decaf::lang::Throwable::getStackTrace () const` [pure virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

Returns

vector containing stack trace strings

Implemented in **decaf::lang::Exception** (p. 1716).

6.813.3.5 `virtual std::string decaf::lang::Throwable::getStackTraceString () const`
[pure virtual]

Gets the stack trace as one contiguous string.

Returns

string with formatted stack trace data

Implemented in **decaf::lang::Exception** (p. 1717).

6.813.3.6 `virtual void decaf::lang::Throwable::initCause (const std::exception *
cause)` [pure virtual]

Initializes the contained cause exception with the one given.

A copy is made to avoid ownership issues.

Parameters

cause The exception that was the cause of this one.

Implemented in **decaf::lang::Exception** (p. 1717).

6.813.3.7 `virtual void decaf::lang::Throwable::printStackTrace () const` [pure
virtual]

Prints the stack trace to std::err.

Implemented in **decaf::lang::Exception** (p. 1717).

6.813.3.8 `virtual void decaf::lang::Throwable::printStackTrace (std::ostream &
stream) const` [pure virtual]

Prints the stack trace to the given output stream.

Parameters

stream the target output stream.

Implemented in **decaf::lang::Exception** (p. 1717).

6.813.3.9 `virtual void decaf::lang::Throwable::setMark (const char * file, const
int lineNumber)` [pure virtual]

Adds a file/line number to the stack trace.

Parameters

file The name of the file calling this method (use `__FILE__`).

lineNumber The line number in the calling file (use `__LINE__`).

Implemented in **decaf::lang::Exception** (p. 1718).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Throwable.h`

6.814 decaf::util::concurrent::TimeoutException Class Reference

```
#include <src/main/decaf/util/concurrent/TimeoutException.h>
```

Inheritance diagram for `decaf::util::concurrent::TimeoutException`:

Public Member Functions

- **TimeoutException** () throw ()
Default Constructor.
- **TimeoutException** (const **decaf::lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **TimeoutException** (const **TimeoutException** &ex) throw ()
Copy Constructor.
- **TimeoutException** (const std::exception *cause) throw ()
Constructor.
- **TimeoutException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **TimeoutException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **TimeoutException** * clone () const
Clones this exception.
- virtual ~**TimeoutException** () throw ()

6.814.1 Constructor & Destructor Documentation

6.814.1.1 decaf::util::concurrent::TimeoutException::TimeoutException () throw () [inline]

Default Constructor.

6.814.1.2 decaf::util::concurrent::TimeoutException::TimeoutException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.814.1.3 decaf::util::concurrent::TimeoutException::TimeoutException (const TimeoutException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex The exception to copy from.

References decaf::lang::Exception::Exception().

6.814.1.4 decaf::util::concurrent::TimeoutException::TimeoutException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.814.1.5 decaf::util::concurrent::TimeoutException::TimeoutException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The string message to report

... list of primitives that are formatted into the message

6.814.1.6 decaf::util::concurrent::TimeoutException::TimeoutException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The string message to report
 ... list of primitives that are formatted into the message

6.814.1.7 `virtual decaf::util::concurrent::TimeoutException::~~TimeoutException () throw () [inline, virtual]`

6.814.2 Member Function Documentation

6.814.2.1 `virtual TimeoutException* decaf::util::concurrent::TimeoutException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new **TimeoutException** (p. 3541) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeoutException.h`

6.815 decaf::util::Timer Class Reference

A facility for threads to schedule tasks for future execution in a background thread.

```
#include <src/main/decaf/util/Timer.h>
```

Public Member Functions

- **Timer** ()
- virtual **~Timer** ()
- void **cancel** ()
Terminates this timer, discarding any currently scheduled tasks.
- `std::size_t` **purge** ()
Removes all canceled tasks from this timer's task queue.
- void **schedule** (**TimerTask** *task, long long delay) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for execution after the specified delay.

- void **schedule** (const **decaf::lang::Pointer**< **TimerTask** > &task, long long delay) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for execution after the specified delay.
- void **schedule** (**TimerTask** *task, const **Date** &time) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for execution at the specified time.
- void **schedule** (const **decaf::lang::Pointer**< **TimerTask** > &task, const **Date** &time) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for execution at the specified time.
- void **schedule** (**TimerTask** *task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.
- void **schedule** (const **decaf::lang::Pointer**< **TimerTask** > &task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.
- void **schedule** (**TimerTask** *task, const **Date** &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.
- void **schedule** (const **decaf::lang::Pointer**< **TimerTask** > &task, const **Date** &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.
- void **scheduleAtFixedRate** (**TimerTask** *task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.
- void **scheduleAtFixedRate** (const **decaf::lang::Pointer**< **TimerTask** > &task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

- `void scheduleAtFixedRate (TimerTask *task, const Date &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

- `void scheduleAtFixedRate (const decaf::lang::Pointer< TimerTask > &task, const Date &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

6.815.1 Detailed Description

A facility for threads to schedule tasks for future execution in a background thread. Tasks may be scheduled for one-time execution, or for repeated execution at regular intervals.

Corresponding to each **Timer** (p. 3543) object is a single background thread that is used to execute all of the timer's tasks, sequentially. **Timer** (p. 3543) tasks should complete quickly. If a timer task takes excessive time to complete, it "hogs" the timer's task execution thread. This can, in turn, delay the execution of subsequent tasks, which may "bunch up" and execute in rapid succession when (and if) the offending task finally completes.

This class is thread-safe: multiple threads can share a single **Timer** (p. 3543) object without the need for external synchronization.

This class does not offer real-time guarantees: it schedules tasks using the `wait(long)` method.

Since

1.0

6.815.2 Constructor & Destructor Documentation

6.815.2.1 `decaf::util::Timer::Timer ()`

6.815.2.2 `virtual decaf::util::Timer::~~Timer () [virtual]`

6.815.3 Member Function Documentation

6.815.3.1 `void decaf::util::Timer::cancel ()`

Terminates this timer, discarding any currently scheduled tasks.

Does not interfere with a currently executing task (if it exists). Once a timer has been terminated, its execution thread terminates gracefully, and no more tasks may be scheduled on it.

Note that calling this method from within the `run` method of a timer task that was invoked by this timer absolutely guarantees that the ongoing task execution is the last task execution that will ever be performed by this timer.

This method may be called repeatedly; the second and subsequent calls have no effect.

6.815.3.2 std::size_t decaf::util::Timer::purge ()

Removes all canceled tasks from this timer's task queue.

Calling this method has no effect on the behavior of the timer, but eliminates the canceled tasks from the queue causing the **Timer** (p. 3543) to destroy the **TimerTask** (p. 3554) pointer it was originally given, the caller should ensure that they no longer have any references to TimerTasks that were previously scheduled.

Most programs will have no need to call this method. It is designed for use by the rare application that cancels a large number of tasks. Calling this method trades time for space: the runtime of the method may be proportional to $n + c \log n$, where n is the number of tasks in the queue and c is the number of canceled tasks.

This method can be called on a **Timer** (p. 3543) object that has no scheduled tasks without error.

Returns

the number of tasks removed from the queue.

6.815.3.3 void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & task, long long delay) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for execution after the specified delay.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

Exceptions

NullPointerException - if the **TimerTask** (p. 3554) value is Null.

IllegalArgumentException - if delay is negative, or delay + System.currentTimeMillis() is negative.

IllegalStateException - if task was already scheduled or cancelled, or timer was cancelled.

6.815.3.4 void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will

generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3554) value is Null.

IllegalArgumentException - if delay is negative, or delay + `System.currentTimeMillis()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.815.3.5 void decaf::util::Timer::schedule ( TimerTask * task,
const Date & firstTime, long long period )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 3554) pointer is considered to be owned by the **Timer** (p. 3543) class once it has been scheduled, the **Timer** (p. 3543) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3543) itself is cancelled. A **TimerTask** (p. 3554) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3543) and the caller should ensure that the **TimerTask** (p. 3554) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3554) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

task - task to be scheduled.

firstTime - First time at which task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3554) value is Null.

IllegalArgumentException - if time.getTime() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.815.3.6 void decaf::util::Timer::schedule (TimerTask * *task*, const Date & *time*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for execution at the specified time.

If the time is in the past, the task is scheduled for immediate execution.

The **TimerTask** (p. 3554) pointer is considered to be owned by the **Timer** (p. 3543) class once it has been scheduled, the **Timer** (p. 3543) will destroy its TimerTask's once they have been cancelled or the **Timer** (p. 3543) itself is cancelled. A **TimerTask** (p. 3554) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3543) and the caller should ensure that the **TimerTask** (p. 3554) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3554) instance are planned.

Parameters

task - task to be scheduled.

time - time at which task is to be executed.

Exceptions

NullPointerException - if the **TimerTask** (p. 3554) value is Null.

IllegalArgumentException - if time.getTime() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.815.3.7 void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & *task*, const Date & *firstTime*, long long *period*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

task - task to be scheduled.

firstTime - First time at which task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3554) value is Null.

IllegalArgumentException - if `time.getTime()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.815.3.8 void decaf::util::Timer::schedule ( TimerTask * task, long long
delay ) throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for execution after the specified delay.

The **TimerTask** (p. 3554) pointer is considered to be owned by the **Timer** (p. 3543) class once it has been scheduled, the **Timer** (p. 3543) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3543) itself is cancelled. A **TimerTask** (p. 3554) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3543) and the caller should ensure that the **TimerTask** (p. 3554) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3554) instance are planned.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

Exceptions

NullPointerException - if the **TimerTask** (p. 3554) value is Null.

IllegalArgumentException - if delay is negative, or delay + `System.currentTimeMillis()` is negative.

IllegalStateException - if task was already scheduled or cancelled, or timer was cancelled.

6.815.3.9 void decaf::util::Timer::schedule (TimerTask * *task*, long long *delay*, long long *period*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay. Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 3554) pointer is considered to be owned by the **Timer** (p. 3543) class once it has been scheduled, the **Timer** (p. 3543) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3543) itself is cancelled. A **TimerTask** (p. 3554) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3543) and the caller should ensure that the **TimerTask** (p. 3554) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3554) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3554) value is Null.

IllegalArgumentException - if delay is negative, or delay + `System.currentTimeMillis()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.815.3.10 void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & *task*, const Date & *time*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for execution at the specified time.

If the time is in the past, the task is scheduled for immediate execution.

Parameters

task - task to be scheduled.

time - time at which task is to be executed.

Exceptions

NullPointerException - if the **TimerTask** (p. 3554) value is Null.

IllegalArgumentException - if time.getTime() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.815.3.11 void decaf::util::Timer::scheduleAtFixedRate ( TimerTask
* task, long long delay, long long period )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3554) pointer is considered to be owned by the **Timer** (p. 3543) class once it has been scheduled, the **Timer** (p. 3543) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3543) itself is cancelled. A **TimerTask** (p. 3554) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3543) and the caller should ensure that the **TimerTask** (p. 3554) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3554) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3554) value is Null.

IllegalArgumentException - if delay is negative, or delay + System.currentTimeMillis() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.815.3.12 `void decaf::util::Timer::scheduleAtFixedRate (const
decaf::lang::Pointer< TimerTask > & task, long long delay, long
long period) throw (decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay. Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3554) value is Null.

IllegalArgumentException - if delay is negative, or delay + `System.currentTimeMillis()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.815.3.13 `void decaf::util::Timer::scheduleAtFixedRate (const
decaf::lang::Pointer< TimerTask > & task, const
Date & firstTime, long long period) throw
(decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

task - task to be scheduled.

firstTime - First time at which task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3554) value is Null.

IllegalArgumentException - if `time.getTime()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.815.3.14 void decaf::util::Timer::scheduleAtFixedRate ( TimerTask
* task, const Date & firstTime, long long period )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3554) pointer is considered to be owned by the **Timer** (p. 3543) class once it has been scheduled, the **Timer** (p. 3543) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3543) itself is cancelled. A **TimerTask** (p. 3554) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3543) and the caller should ensure that the **TimerTask** (p. 3554) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3554) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a

fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

- task*** - task to be scheduled.
- firstTime*** - First time at which task is to be executed.
- period*** - time in milliseconds between successive task executions.

Exceptions

- NullPointerException*** - if the **TimerTask** (p. 3554) value is Null.
- IllegalArgumentException*** - if time.getTime() is negative.
- IllegalStateException*** - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Timer.h**

6.816 decaf::util::TimerTask Class Reference

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3543).

```
#include <src/main/decaf/util/TimerTask.h>
```

Inheritance diagram for decaf::util::TimerTask:

Public Member Functions

- **TimerTask** ()
- virtual **~TimerTask** ()
- bool **cancel** ()
Cancels this timer task.
- long long **scheduledExecutionTime** () const
Returns the scheduled execution time of the most recent actual execution of this task.

Protected Member Functions

- bool **isScheduled** () const
- void **setScheduledTime** (long long time)
- long long **getWhen** () const

Friends

- class `Timer`
- class `TimerImpl`
- class `decaf::internal::util::TimerTaskHeap`

6.816.1 Detailed Description

A Base class for a task object that can be scheduled for one-time or repeated execution by a `Timer` (p. 3543).

Since

1.0

6.816.2 Constructor & Destructor Documentation

6.816.2.1 `decaf::util::TimerTask::TimerTask ()`

6.816.2.2 `virtual decaf::util::TimerTask::~~TimerTask () [inline, virtual]`

6.816.3 Member Function Documentation

6.816.3.1 `bool decaf::util::TimerTask::cancel ()`

Cancels this timer task.

If the task has been scheduled for one-time execution and has not yet run, or has not yet been scheduled, it will never run. If the task has been scheduled for repeated execution, it will never run again. (If the task is running when this call occurs, the task will run to completion, but will never run again.)

Note that calling this method from within the run method of a repeating timer task absolutely guarantees that the timer task will not run again.

This method may be called repeatedly; the second and subsequent calls have no effect.

Returns

true if this task is scheduled for one-time execution and has not yet run, or this task is scheduled for repeated execution. Returns false if the task was scheduled for one-time execution and has already run, or if the task was never scheduled, or if the task was already canceled. (Loosely speaking, this method returns true if it prevents one or more scheduled executions from taking place.)

6.816.3.2 `long long decaf::util::TimerTask::getWhen () const [protected]`

6.816.3.3 `bool decaf::util::TimerTask::isScheduled () const [protected]`

6.816.3.4 `long long decaf::util::TimerTask::scheduledExecutionTime () const`

Returns the scheduled execution time of the most recent actual execution of this task.

(If this method is invoked while task execution is in progress, the return value is the scheduled execution time of the ongoing task execution.)

This method is typically invoked from within a task's run method, to determine whether the current execution of the task is sufficiently timely to warrant performing the scheduled activity:

```
void run() (p. 3112) { if( System::currentTimeMillis() - scheduledExecutionTime() (p. 3555)
>= MAX_TARDINESS) return; // Too late; skip this execution. // Perform the task }
```

This method is typically not used in conjunction with fixed-delay execution repeating tasks, as their scheduled execution times are allowed to drift over time, and so are not terribly significant.

Returns

the time at which the most recent execution of this task was scheduled to occur, in the format returned by **Date.getTime()** (p. 1561). The return value is undefined if the task has yet to commence its first execution.

6.816.3.5 void decaf::util::TimerTask::setScheduledTime (long long *time*)
[protected]

6.816.4 Friends And Related Function Documentation

6.816.4.1 friend class decaf::internal::util::TimerTaskHeap [friend]

6.816.4.2 friend class Timer [friend]

6.816.4.3 friend class TimerImpl [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**TimerTask.h**

6.817 decaf::internal::util::TimerTaskHeap Class Reference

A Binary Heap implemented specifically for the Timer class in Decaf Util.

```
#include <src/main/decaf/internal/util/TimerTaskHeap.h>
```

Public Member Functions

- **TimerTaskHeap** ()
- virtual **~TimerTaskHeap** ()
- **Pointer< TimerTask > peek** ()

Peeks at the Head of the Heap, returns the task with the nearest scheduled run time.

- bool **isEmpty** () const
- std::size_t **size** () const
- void **insert** (const **Pointer< TimerTask >** &task)

Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.

- void **remove** (std::size_t pos)

Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.

- void **reset** ()

Clear all contents from the heap.

- void **adjustMinimum** ()

Resorts the heap starting at the top.

- std::size_t **deleteIfCancelled** ()

Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a cancellation of a large number of tasks the user can perform this purge.

- std::size_t **find** (const **Pointer**< **TimerTask** > &task) const

Searches the heap for the specified TimerTask element and returns its position in the heap.

6.817.1 Detailed Description

A Binary Heap implemented specifically for the Timer class in Decaf Util.

Since

1.0

6.817.2 Constructor & Destructor Documentation

6.817.2.1 **decaf::internal::util::TimerTaskHeap::TimerTaskHeap** ()

6.817.2.2 **virtual decaf::internal::util::TimerTaskHeap::~~TimerTaskHeap** ()
[virtual]

6.817.3 Member Function Documentation

6.817.3.1 **void decaf::internal::util::TimerTaskHeap::adjustMinimum** ()

Resorts the heap starting at the top.

6.817.3.2 **std::size_t decaf::internal::util::TimerTaskHeap::deleteIfCancelled** ()

Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a cancellation of a large number of tasks the user can perform this purge.

Returns

the number of task that were removed from the heap because they were cancelled.

6.817.3.3 `std::size_t decaf::internal::util::TimerTaskHeap::find (const Pointer< TimerTask > & task) const`

Searches the heap for the specified TimerTask element and returns its position in the heap.

Returns the unsigned equivalent of -1 if the element is not found.

Returns

the position in the Heap where the Task is stored, or npos.

6.817.3.4 `void decaf::internal::util::TimerTaskHeap::insert (const Pointer< TimerTask > & task)`

Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.

Parameters

task The TimerTask to insert into the heap.

6.817.3.5 `bool decaf::internal::util::TimerTaskHeap::isEmpty () const`**Returns**

true if the heap is empty.

6.817.3.6 `Pointer<TimerTask> decaf::internal::util::TimerTaskHeap::peek ()`

Peaks at the Head of the Heap, returns the task with the nearest scheduled run time.

Returns

The TimerTask that is scheduled to be executed next if the Heap is empty a Null Pointer value is returned.

6.817.3.7 `void decaf::internal::util::TimerTaskHeap::remove (std::size_t pos)`

Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.

Parameters

pos The position at which to remove the TimerTask and begin a resort of the heap.

6.817.3.8 `void decaf::internal::util::TimerTaskHeap::reset ()`

Clear all contents from the heap.

6.817.3.9 `std::size_t decaf::internal::util::TimerTaskHeap::size () const`

Returns

the size of the heap.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/TimerTaskHeap.h`

6.818 `decaf::util::concurrent::TimeUnit` Class Reference

A **TimeUnit** (p. 3559) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

```
#include <src/main/decaf/util/concurrent/TimeUnit.h>
```

Inheritance diagram for `decaf::util::concurrent::TimeUnit`:

Public Member Functions

- virtual `~TimeUnit ()`
- long long **convert** (long long sourceDuration, const **TimeUnit** &sourceUnit) const
Convert the given time duration in the given unit to this unit.
- long long **toNanos** (long long duration) const
Equivalent to `NANOSECONDS.convert(duration, this)`.
- long long **toMicros** (long long duration) const
Equivalent to `MICROSECONDS.convert(duration, this)`.
- long long **toMillis** (long long duration) const
Equivalent to `MILLISECONDS.convert(duration, this)`.
- long long **toSeconds** (long long duration) const
Equivalent to `SECONDS.convert(duration, this)`.
- long long **toMinutes** (long long duration) const
Equivalent to `MINUTES.convert(duration, this)`.
- long long **toHours** (long long duration) const
Equivalent to `HOURS.convert(duration, this)`.
- long long **toDays** (long long duration) const
Equivalent to `DAYS.convert(duration, this)`.
- void **timedWait** (**Synchronizable** *obj, long long timeout) const throw (`decaf::lang::exceptions::InterruptedException`, `decaf::lang::exceptions::NullPointerException`)

Perform a timed `Object.wait` using this time unit.

- void **timedJoin** (decaf::lang::Thread *thread, long long timeout) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException)

Perform a timed `Thread.join` using this time unit.

- void **sleep** (long long timeout) const throw (decaf::lang::exceptions::InterruptedException)

Perform a `Thread.sleep` using this unit.

- virtual std::string **toString** () const

*Converts the **TimeUnit** (p. 3559) type to the Name of the **TimeUnit** (p. 3559).*

- virtual int **compareTo** (const TimeUnit &value) const

Compares this object with the specified object for order.

- virtual bool **equals** (const TimeUnit &value) const

- virtual bool **operator==** (const TimeUnit &value) const

Compares equality between this object and the one passed.

- virtual bool **operator<** (const TimeUnit &value) const

Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static const **TimeUnit** & **valueOf** (const std::string &name) throw (decaf::lang::exceptions::IllegalArgumentException)

*Returns the **TimeUnit** (p. 3559) constant of this type with the specified name.*

Static Public Attributes

- static const **TimeUnit** **NANOSECONDS**

*The Actual **TimeUnit** (p. 3559) enumerations.*

- static const **TimeUnit** **MICROSECONDS**

- static const **TimeUnit** **MILLISECONDS**

- static const **TimeUnit** **SECONDS**

- static const **TimeUnit** **MINUTES**

- static const **TimeUnit** **HOURS**

- static const **TimeUnit** **DAYS**

- static const **TimeUnit** *const **values** []

*The An Array of **TimeUnit** (p. 3559) Instances.*

Protected Member Functions

- **TimeUnit** (int index, const std::string &name)

Hidden Constructor, this class can not be instantiated directly.

6.818.1 Detailed Description

A **TimeUnit** (p. 3559) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units. A **TimeUnit** (p. 3559) does not maintain time information, but only helps organize and use time representations that may be maintained separately across various contexts. A nanosecond is defined as one thousandth of a microsecond, a microsecond as one thousandth of a millisecond, a millisecond as one thousandth of a second, a minute as sixty seconds, an hour as sixty minutes, and a day as twenty four hours.

A **TimeUnit** (p. 3559) is mainly used to inform time-based methods how a given timing parameter should be interpreted. For example, the following code will timeout in 50 milliseconds if the lock is not available:

```
Lock (p. 2228) lock = ...; if ( lock.tryLock( 50, TimeUnit.MILLISECONDS (p. 3567) ) ) ...
```

while this code will timeout in 50 seconds:

```
Lock (p. 2228) lock = ...; if ( lock.tryLock( 50, TimeUnit.SECONDS (p. 3568) ) ) ...
```

Note however, that there is no guarantee that a particular timeout implementation will be able to notice the passage of time at the same granularity as the given **TimeUnit** (p. 3559).

6.818.2 Constructor & Destructor Documentation

6.818.2.1 `decaf::util::concurrent::TimeUnit::TimeUnit (int index, const std::string & name)` [protected]

Hidden Constructor, this class can not be instantiated directly.

Parameters

index - Index into the Time Unit set.

name - Name of the unit type being represented.

6.818.2.2 `virtual decaf::util::concurrent::TimeUnit::~~TimeUnit ()` [inline, virtual]

6.818.3 Member Function Documentation

6.818.3.1 `virtual int decaf::util::concurrent::TimeUnit::compareTo (const TimeUnit & value) const` [virtual]

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation `sgn(expression)` designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure `sgn(x.compareTo(y)) == -sgn(y.compareTo(x))` for all `x` and `y`. (This implies that `x.compareTo(y)` must throw an exception iff `y.compareTo(x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `(x.compareTo(y)>0 && y.compareTo(z)>0)` implies `x.compareTo(z)>0`.

Finally, the implementor must ensure that `x.compareTo(y)==0` implies that `sgn(x.compareTo(z)) == sgn(y.compareTo(z))`, for all `z`.

It is strongly recommended, but not strictly required that `(x.compareTo(y)==0) == (x.equals(y))`. Generally speaking, any class that implements the `Comparable` interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

Parameters

value - the Object to be compared.

Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.818.3.2 long long decaf::util::concurrent::TimeUnit::convert (long long *sourceDuration*, const TimeUnit & *sourceUnit*) const

Convert the given time duration in the given unit to this unit.

Conversions from finer to coarser granularities truncate, so lose precision. For example converting 999 milliseconds to seconds results in 0. Conversions from coarser to finer granularities with arguments that would numerically overflow saturate to `Long.MIN_VALUE` if negative or `Long.MAX_VALUE` if positive.

For example, to convert 10 minutes to milliseconds, use: `TimeUnit.MILLISECONDS.convert(10L, TimeUnit.MINUTES)` (p. 3567)

Parameters

sourceDuration - Duration value to convert.

sourceUnit - Unit type of the source duration.

Returns

the converted duration in this unit, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

6.818.3.3 virtual bool decaf::util::concurrent::TimeUnit::equals (const TimeUnit & *value*) const [virtual]

Returns

true if this value is considered equal to the passed value.

6.818.3.4 `virtual bool decaf::util::concurrent::TimeUnit::operator< (const
TimeUnit & value) const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.818.3.5 `virtual bool decaf::util::concurrent::TimeUnit::operator== (const
TimeUnit & value) const [virtual]`

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.818.3.6 `void decaf::util::concurrent::TimeUnit::sleep (long long timeout) const
throw (decaf::lang::exceptions::InterruptedException)`

Perform a `Thread.sleep` using this unit.

This is a convenience method that converts time arguments into the form required by the `Thread.sleep` method.

Parameters

timeout the minimum time to sleep

See also

`Thread::sleep`

6.818.3.7 `void decaf::util::concurrent::TimeUnit::timedJoin (decaf::lang::Thread * thread, long long timeout)
throw (decaf::lang::exceptions::InterruptedException,
decaf::lang::exceptions::NullPointerException)`

Perform a timed `Thread.join` using this time unit.

This is a convenience method that converts time arguments into the form required by the `Thread.join` method.

Parameters

thread the thread to wait for

timeout the maximum time to wait

Exceptions

InterruptedException if interrupted while waiting.

NullPointerException if the thread object is null.

See also

Thread::join(long long, long long)

6.818.3.8 void decaf::util::concurrent::TimeUnit::timedWait (Synchronizable * *obj*, long long *timeout*) const
throw (decaf::lang::exceptions::InterruptedException,
decaf::lang::exceptions::NullPointerException)

Perform a timed Object.wait using this time unit.

This is a convenience method that converts timeout arguments into the form required by the Object.wait method.

For example, you could implement a blocking poll method (see **BlockingQueue.poll** (p. 779)) using:

```
Object poll( long long timeout, const TimeUnit& unit )
    throw( InterruptedException ) {

    while( empty ) {
        unit.timedWait(this, timeout);
        ...
    }
}
```

Parameters

obj the object to wait on

timeout the maximum time to wait.

Exceptions

InterruptedException if interrupted while waiting.

NullPointerException if the **Synchronizable** (p. 3461) object is null.

See also

Synchronizable::wait(long long, long long)

6.818.3.9 long long decaf::util::concurrent::TimeUnit::toDays (long long *duration*) const [inline]

Equivalent to DAYS.convert(duration, this).

Parameters

duration the duration

Returns

the converted duration.

See also

`convert` (p. 3562)

6.818.3.10 `long long decaf::util::concurrent::TimeUnit::toHours (long long
duration) const [inline]`

Equivalent to `HOURS.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration.

See also

`convert` (p. 3562)

6.818.3.11 `long long decaf::util::concurrent::TimeUnit::toMicros (long long
duration) const [inline]`

Equivalent to `MICROSECONDS.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also

`convert` (p. 3562)

6.818.3.12 `long long decaf::util::concurrent::TimeUnit::toMillis (long long
duration) const [inline]`

Equivalent to `MILLISECONDS.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration, or Long.MIN_VALUE if conversion would negatively overflow, or Long.MAX_VALUE if it would positively overflow.

See also

`convert` (p. 3562)

6.818.3.13 `long long decaf::util::concurrent::TimeUnit::toMinutes (long long duration) const [inline]`

Equivalent to `MINUTES.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration.

See also

`convert` (p. 3562)

6.818.3.14 `long long decaf::util::concurrent::TimeUnit::toNanos (long long duration) const [inline]`

Equivalent to `NANOSECONDS.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration, or Long.MIN_VALUE if conversion would negatively overflow, or Long.MAX_VALUE if it would positively overflow.

See also

`convert` (p. 3562)

6.818.3.15 `long long decaf::util::concurrent::TimeUnit::toSeconds (long long duration) const [inline]`

Equivalent to `SECONDS.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration.

See also

`convert` (p. 3562)

6.818.3.16 `virtual std::string decaf::util::concurrent::TimeUnit::toString () const`
[virtual]

Converts the **TimeUnit** (p. 3559) type to the Name of the **TimeUnit** (p. 3559).

Returns

String name of the **TimeUnit** (p. 3559)

6.818.3.17 `static const TimeUnit& decaf::util::concurrent::TimeUnit::valueOf`
`(const std::string & name) throw (de-`
`cafe::lang::exceptions::IllegalArgumentException)`
[static]

Returns the **TimeUnit** (p. 3559) constant of this type with the specified name.

The string must match exactly an identifier used to declare an **TimeUnit** (p. 3559) constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters

name The Name of the **TimeUnit** (p. 3559) constant to be returned.

Returns

A constant reference to the **TimeUnit** (p. 3559) Constant with the given name.

Exceptions

IllegalArgumentException if this enum type has no constant with the specified name

6.818.4 Field Documentation

6.818.4.1 `const TimeUnit decaf::util::concurrent::TimeUnit::DAYS` [static]

6.818.4.2 `const TimeUnit decaf::util::concurrent::TimeUnit::HOURS` [static]

6.818.4.3 `const TimeUnit decaf::util::concurrent::TimeUnit::MICROSECONDS`
[static]

6.818.4.4 `const TimeUnit decaf::util::concurrent::TimeUnit::MILLISECONDS`
[static]

6.818.4.5 `const TimeUnit decaf::util::concurrent::TimeUnit::MINUTES` [static]

6.818.4.6 `const TimeUnit decaf::util::concurrent::TimeUnit::NANOSECONDS`
[static]

The Actual **TimeUnit** (p. 3559) enumerations.

6.818.4.7 `const TimeUnit decaf::util::concurrent::TimeUnit::SECONDS` [static]

6.818.4.8 `const TimeUnit* const decaf::util::concurrent::TimeUnit::values[]`
[static]

The An Array of **TimeUnit** (p. 3559) Instances.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeUnit.h`

6.819 cms::Topic Class Reference

An interface encapsulating a provider-specific topic name.

```
#include <src/main/cms/Topic.h>
```

Inheritance diagram for cms::Topic:

Public Member Functions

- `virtual ~Topic ()`
- `virtual std::string getTopicName () const =0 throw (CMSEException)`
Gets the name of this topic.

6.819.1 Detailed Description

An interface encapsulating a provider-specific topic name. A **Topic** (p. 3568) is a Publish / Subscribe type **Destination** (p. 1610). All Messages sent to a **Topic** (p. 3568) are broadcast to all Subscribers of that **Topic** (p. 3568) unless the Subscriber defines a **Message** (p. 2375) selector that filters out that **Message** (p. 2375).

Since

1.0

6.819.2 Constructor & Destructor Documentation

6.819.2.1 `virtual cms::Topic::~~Topic ()` [inline, virtual]

6.819.3 Member Function Documentation

6.819.3.1 `virtual std::string cms::Topic::getTopicName () const throw (CMSEException)` [pure virtual]

Gets the name of this topic.

Returns

The topic name.

Exceptions

CMSEException (p. 1074) - If an internal error occurs.

Implemented in `activemq::commands::ActiveMQTopic` (p. 636).

The documentation for this class was generated from the following file:

- `src/main/cms/Topic.h`

6.820 `activemq::state::Tracked` Class Reference

```
#include <src/main/activemq/state/Tracked.h>
```

Inheritance diagram for `activemq::state::Tracked`:

Public Member Functions

- `Tracked ()`
- `Tracked (const Pointer< decaf::lang::Runnable > &runnable)`
- `virtual ~Tracked ()`
- `void onResponse ()`
- `bool isWaitingForResponse () const`

6.820.1 Constructor & Destructor Documentation

6.820.1.1 `activemq::state::Tracked::Tracked () [inline]`

6.820.1.2 `activemq::state::Tracked::Tracked (const Pointer< decaf::lang::Runnable > & runnable)`

6.820.1.3 `virtual activemq::state::Tracked::~~Tracked () [inline, virtual]`

6.820.2 Member Function Documentation

6.820.2.1 `bool activemq::state::Tracked::isWaitingForResponse () const [inline]`

6.820.2.2 `void activemq::state::Tracked::onResponse ()`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/Tracked.h`

6.821 `activemq::commands::TransactionId` Class Reference

```
#include <src/main/activemq/commands/TransactionId.h>
```

Inheritance diagram for activemq::commands::TransactionId:

Public Types

- typedef decaf::lang::PointerComparator< TransactionId > COMPARATOR

Public Member Functions

- TransactionId ()
- TransactionId (const TransactionId &other)
- virtual ~TransactionId ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual TransactionId * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const DataStructure *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const DataStructure *value) const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual int **compareTo** (const TransactionId &value) const
- virtual bool **equals** (const TransactionId &value) const
- virtual bool **operator==** (const TransactionId &value) const
- virtual bool **operator<** (const TransactionId &value) const
- TransactionId & **operator=** (const TransactionId &other)

Static Public Attributes

- static const unsigned char **ID_TRANSACTIONID** = 0

6.821.1 Member Typedef Documentation

6.821.1.1 `typedef decaf::lang::PointerComparator<TransactionId>
activemq::commands::TransactionId::COMPARATOR`

6.821.2 Constructor & Destructor Documentation

6.821.2.1 `activemq::commands::TransactionId::TransactionId ()`

6.821.2.2 `activemq::commands::TransactionId::TransactionId (const
TransactionId & other)`

6.821.2.3 `virtual activemq::commands::TransactionId::~~TransactionId ()
[virtual]`

6.821.3 Member Function Documentation

6.821.3.1 `virtual TransactionId* ac-
tivemq::commands::TransactionId::cloneDataStructure (
) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.821.3.2 `virtual int activemq::commands::TransactionId::compareTo (const
TransactionId & value) const [virtual]`

6.821.3.3 `virtual void activemq::commands::TransactionId::copyDataStructure (
const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.821.3.4 `virtual bool activemq::commands::TransactionId::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.821.3.5 virtual bool activemq::commands::TransactionId::equals (const TransactionId & *value*) const [virtual]

6.821.3.6 virtual unsigned char activemq::commands::TransactionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaller share.

Returns

new **DataStructure** (p. 1553) type copy.

6.821.3.7 virtual bool activemq::commands::TransactionId::operator< (const TransactionId & *value*) const [virtual]

6.821.3.8 TransactionId& activemq::commands::TransactionId::operator= (const TransactionId & *other*)

6.821.3.9 virtual bool activemq::commands::TransactionId::operator== (const TransactionId & *value*) const [virtual]

6.821.3.10 virtual std::string activemq::commands::TransactionId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.821.4 Field Documentation

6.821.4.1 const unsigned char activemq::commands::TransactionId::ID_TRANSACTIONID = 0 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**TransactionId.h**

6.822 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3572).

#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.822.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3572). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.822.2 Constructor & Destructor Documentation

6.822.2.1 `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.822.2.2 `virtual activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

6.822.3 Member Function Documentation

6.822.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2218), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3791).

6.822.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2218), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3791).

```
6.822.3.3  virtual int ac-
            tivemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat,  commands::DataStructure
            * dataStructure,  utils::BooleanStream * bs ) throw (
            decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2218), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3791).

```
6.822.3.4  virtual void ac-
            tivemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat,  commands::DataStructure
            * dataStructure,  decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

6.823 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller Class Reference 3583

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2219), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3792).

6.822.3.5 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2219), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3792).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h`

6.823 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for `TransactionIdMarshaller` (p. 3576).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller`:

Public Member Functions

- `TransactionIdMarshaller ()`
- `virtual ~TransactionIdMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.823.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3576). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.823.2 Constructor & Destructor Documentation

6.823.2.1 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::TransactionIdMarshaller () [inline]

6.823.2.2 virtual activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::~~TransactionIdMarshaller () [inline, virtual]

6.823.3 Member Function Documentation

6.823.3.1 virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1518).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2226), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3783).

6.823.3.2 virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2226), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3783).

6.823.3.3 virtual int activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2226), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3783).

6.823.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2227), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3784).

6.823.3.5 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2227), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h`

6.824 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3580).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller**:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.824.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3580). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.824.2 Constructor & Destructor Documentation

6.824.2.1 `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::TransactionIdMarshaller ()` [inline]

6.824.2.2 `virtual activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::~~TransactionIdMarshaller ()` [inline, virtual]

6.824.3 Member Function Documentation

6.824.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 2210), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3775).

6.824.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2210), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 3775).

6.824.3.3 `virtual int activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2210), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 3775).

6.824.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 2211), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3776).

```
6.824.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 2211), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3776).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h`

6.825 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for `TransactionIdMarshaller` (p. 3583).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller`:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.825.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3583). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.825.2 Constructor & Destructor Documentation

6.825.2.1 `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::TransactionIdMarshaller () [inline]`

6.825.2.2 `virtual activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::~~TransactionIdMarshaller () [inline, virtual]`

6.825.3 Member Function Documentation

6.825.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2214), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3787).

6.825.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2214), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3787).

6.825.3.3 `virtual int activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2214), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3787).

6.825.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2215), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3788).

```
6.825.3.5 virtual void ac-
      tivemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightUnmarshal
      ( OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataInputStream * dataIn,
      utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2215), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3788).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h`

6.826 `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `TransactionIdMarshaller` (p. 3587).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller`:

Public Member Functions

- `TransactionIdMarshaller ()`
- `virtual ~TransactionIdMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.826.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3587). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.826.2 Constructor & Destructor Documentation

6.826.2.1 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::TransactionIdMarshaller () [inline]

6.826.2.2 virtual activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::~~TransactionIdMarshaller () [inline, virtual]

6.826.3 Member Function Documentation

6.826.3.1 virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2222), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3779).

6.826.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2222), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3779).

6.826.3.3 `virtual int activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2222), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3779).

6.826.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2223), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3780).

6.826.3.5 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2223), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 3780).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h`

6.827 **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3591).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller**:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.827.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3591). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.827.2 Constructor & Destructor Documentation

6.827.2.1 `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::TransactionIdMarshaller ()` [inline]

6.827.2.2 `virtual activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::~~TransactionIdMarshaller ()` [inline, virtual]

6.827.3 Member Function Documentation

6.827.3.1 `virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1518).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2206), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 3771).

6.827.3.2 `virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller` (p. 2206), and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 3771).

6.827.3.3 `virtual int ac-`
`tivemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightMarshal1`
`(OpenWireFormat * wireFormat, commands::DataStructure`
`* dataStructure, utils::BooleanStream * bs) throw (`
`decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller` (p. 2206), and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 3771).

6.827.3.4 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightMarshal2`
`(OpenWireFormat * wireFormat, commands::DataStructure`
`* dataStructure, decaf::io::DataOutputStream * dataOut,`
`utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2207), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 3772).

```
6.827.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2207), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 3772).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h

6.828 activemq::commands::TransactionInfo Class Reference

```
#include <src/main/activemq/commands/TransactionInfo.h>
```

Inheritance diagram for **activemq::commands::TransactionInfo**:

Public Member Functions

- **TransactionInfo** ()

- virtual `~TransactionInfo ()`
- virtual unsigned char `getDataStructureType ()` const
Get the unique identifier that this object and its own Marshaler share.
- virtual `TransactionInfo * cloneDataStructure ()` const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void `copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string `toString ()` const
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- virtual bool `equals (const DataStructure *value)` const
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual const `Pointer< ConnectionId > & getConnectionId ()` const
- virtual `Pointer< ConnectionId > & getConnectionId ()`
- virtual void `setConnectionId (const Pointer< ConnectionId > &connectionId)`
- virtual const `Pointer< TransactionId > & getTransactionId ()` const
- virtual `Pointer< TransactionId > & getTransactionId ()`
- virtual void `setTransactionId (const Pointer< TransactionId > &transactionId)`
- virtual unsigned char `getType ()` const
- virtual void `setType (unsigned char type)`
- virtual bool `isTransactionInfo ()` const
- virtual `Pointer< Command > visit (activemq::state::CommandVisitor *visitor)`
`throw (exceptions::ActiveMQException)`
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char `ID_TRANSACTIONINFO = 7`

Protected Attributes

- `Pointer< ConnectionId > connectionId`
- `Pointer< TransactionId > transactionId`
- unsigned char `type`

6.828.1 Constructor & Destructor Documentation

6.828.1.1 `activemq::commands::TransactionInfo::TransactionInfo ()`

6.828.1.2 `virtual activemq::commands::TransactionInfo::~~TransactionInfo ()`
[virtual]

6.828.2 Member Function Documentation

6.828.2.1 `virtual TransactionInfo* activemq::commands::TransactionInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.828.2.2 `virtual void activemq::commands::TransactionInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.828.2.3 `virtual bool activemq::commands::TransactionInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

- 6.828.2.4** `virtual const Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId () const [virtual]`
- 6.828.2.5** `virtual Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId () [virtual]`
- 6.828.2.6** `virtual unsigned char activemq::commands::TransactionInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSetructure** (p. 1553) type copy.

Implements **activemq::commands::DataSetructure** (p. 1557).

- 6.828.2.7** `virtual const Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId () const [virtual]`
- 6.828.2.8** `virtual Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId () [virtual]`
- 6.828.2.9** `virtual unsigned char activemq::commands::TransactionInfo::getType () const [virtual]`
- 6.828.2.10** `virtual bool activemq::commands::TransactionInfo::isTransactionInfo () const [inline, virtual]`

Returns

an answer of true to the **isTransactionInfo()** (p. 3597) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

- 6.828.2.11** `virtual void activemq::commands::TransactionInfo::setConnectionId (const Pointer< ConnectionId > & connectionId) [virtual]`
- 6.828.2.12** `virtual void activemq::commands::TransactionInfo::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`
- 6.828.2.13** `virtual void activemq::commands::TransactionInfo::setType (unsigned char type) [virtual]`
- 6.828.2.14** `virtual std::string activemq::commands::TransactionInfo::toString () const [virtual]`

Returns a string containing the information for this **DataSetructure** (p. 1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

6.828.2.15 `virtual Pointer<Command> activemq::commands::TransactionInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1112).

6.828.3 Field Documentation

6.828.3.1 `Pointer<ConnectionId> activemq::commands::TransactionInfo::connectionId [protected]`

6.828.3.2 `const unsigned char activemq::commands::TransactionInfo::ID_TRANSACTIONINFO = 7 [static]`

6.828.3.3 `Pointer<TransactionId> activemq::commands::TransactionInfo::transactionId [protected]`

6.828.3.4 `unsigned char activemq::commands::TransactionInfo::type [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/TransactionInfo.h`

6.829 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3598).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller**:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.829.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3598). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.829.2 Constructor & Destructor Documentation

6.829.2.1 `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.829.2.2 `virtual activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.829.3 Member Function Documentation

6.829.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.829.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.829.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 722).

```

6.829.3.4 virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 723).

```

6.829.3.5 virtual int activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 724).

```

6.829.3.6 virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 726).

```
6.829.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 727).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**TransactionInfoMarshaller.h**

6.830 **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3602).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller**:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.830.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3602). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.830.2 Constructor & Destructor Documentation

6.830.2.1 `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.830.2.2 `virtual activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.830.3 Member Function Documentation

6.830.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.830.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.830.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 716).

6.830.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 717).

6.830.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 718).

6.830.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 719).

```
6.830.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 720).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h

6.831 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3606).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller**:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.831.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3606). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.831.2 Constructor & Destructor Documentation

6.831.2.1 `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.831.2.2 `virtual activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.831.3 Member Function Documentation

6.831.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.831.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.831.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 702).

6.831.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 703).

6.831.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 704).

6.831.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 706).

```
6.831.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 707).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h`

6.832 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for `TransactionInfoMarshaller` (p. 3610).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller`:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.832.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3610). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.832.2 Constructor & Destructor Documentation

6.832.2.1 `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::TransactionInfoMarshaller() [inline]`

6.832.2.2 `virtual
activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::~~TransactionInfoMarshaller() [inline, virtual]`

6.832.3 Member Function Documentation

6.832.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::createObject() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.832.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.832.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 729).

```

6.832.3.4 virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 730).

```

6.832.3.5 virtual int activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 731).

```

6.832.3.6 virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 732).

6.832.3.7 virtual void ac-
 tivemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 733).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h

6.833 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3614).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller**:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.833.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3614). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.833.2 Constructor & Destructor Documentation

6.833.2.1 `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.833.2.2 `virtual activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.833.3 Member Function Documentation

6.833.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.833.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.833.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 709).

```

6.833.3.4 virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 710).

```

6.833.3.5 virtual int activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 711).

```

6.833.3.6 virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 712).

6.833.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h`

6.834 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3618).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h>`

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller**:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.834.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3618). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.834.2 Constructor & Destructor Documentation

6.834.2.1 `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.834.2.2 `virtual activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.834.3 Member Function Documentation

6.834.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.834.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.834.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 736).

6.834.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 737).

6.834.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 738).

6.834.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 739).

6.834.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 740).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h`

6.835 activemq::state::TransactionState Class Reference

```
#include <src/main/activemq/state/TransactionState.h>
```

Public Member Functions

- `TransactionState (const Pointer< TransactionId > &id)`
- `virtual ~TransactionState ()`
- `std::string toString () const`
- `void addCommand (const Pointer< Command > &operation)`

- void **checkShutdown** () const
- void **shutdown** ()
- const **StlList**< **Pointer**< **Command** > > & **getCommands** () const
- const **Pointer**< **TransactionId** > & **getId** () const
- void **setPrepared** (bool prepared)
- bool **isPrepared** () const
- void **setPreparedResult** (int preparedResult)
- int **getPreparedResult** () const
- void **addProducerState** (const **Pointer**< **ProducerState** > &producerState)
- std::vector< **Pointer**< **ProducerState** > > **getProducerStates** ()

6.835.1 Constructor & Destructor Documentation

6.835.1.1 `activemq::state::TransactionState::TransactionState (const Pointer< TransactionId > & id)`

6.835.1.2 `virtual activemq::state::TransactionState::~~TransactionState ()`
[virtual]

6.835.2 Member Function Documentation

6.835.2.1 `void activemq::state::TransactionState::addCommand (const Pointer< Command > & operation)`

6.835.2.2 `void activemq::state::TransactionState::addProducerState (const Pointer< ProducerState > & producerState)`

6.835.2.3 `void activemq::state::TransactionState::checkShutdown () const`

6.835.2.4 `const StlList< Pointer<Command> >& activemq::state::TransactionState::getCommands ()`
const [inline]

6.835.2.5 `const Pointer<TransactionId>& activemq::state::TransactionState::getId () const` [inline]

6.835.2.6 `int activemq::state::TransactionState::getPreparedResult () const`
[inline]

6.835.2.7 `std::vector< Pointer<ProducerState> > activemq::state::TransactionState::getProducerStates ()`

6.835.2.8 `bool activemq::state::TransactionState::isPrepared () const` [inline]

6.835.2.9 `void activemq::state::TransactionState::setPrepared (bool prepared)`
[inline]

6.835.2.10 `void activemq::state::TransactionState::setPreparedResult (int preparedResult)` [inline]

6.835.2.11 `void activemq::state::TransactionState::shutdown ()` [inline]

6.835.2.12 `std::string activemq::state::TransactionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/TransactionState.h`

6.836 decaf::internal::util::concurrent::Transferer< E > Class Template Reference

Shared internal API for dual stacks and queues.

```
#include <src/main/decaf/internal/util/concurrent/Transferer.h>
```

Inheritance diagram for decaf::internal::util::concurrent::Transferer< E >:

6.836.1 Detailed Description

```
template<typename E> class decaf::internal::util::concurrent::Transferer< E >
```

Shared internal API for dual stacks and queues.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**Transferer.h**

6.837 decaf::internal::util::concurrent::TransferQueue< E > Class Template Reference

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

```
#include <src/main/decaf/internal/util/concurrent/TransferQueue.h>
```

Inheritance diagram for decaf::internal::util::concurrent::TransferQueue< E >:

Public Member Functions

- **TransferQueue** ()
*Node class for **TransferQueue** (p. 3625).*
- virtual ~**TransferQueue** ()
- virtual void **transfer** (E *e, bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)
Performs a put.
- virtual E * **transfer** (bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)
Performs a take.

6.837.1 Detailed Description

template<typename E> class decaf::internal::util::concurrent::TransferQueue< E >

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using nodes within nodes rather than marked pointers. The algorithm is a little simpler than that for stacks because fulfillers do not need explicit nodes, and matching is done by CAS'ing QNode.item field from non-null to null (for put) or vice versa (for take).

6.837.2 Constructor & Destructor Documentation

6.837.2.1 template<typename E > decaf::internal::util::concurrent::TransferQueue< E >::TransferQueue () [inline]

Node class for **TransferQueue** (p. 3625).

Tries to cancel by CAS'ing ref to NULL if that succeeds then we mark as cancelled. Returns true if this node is known to be off the queue because its next pointer has been forgotten due to an advanceHead operation.

6.837.2.2 template<typename E > virtual
decaf::internal::util::concurrent::TransferQueue< E
>::~~TransferQueue () [inline, virtual]

6.837.3 Member Function Documentation

6.837.3.1 template<typename E > virtual void
decaf::internal::util::concurrent::TransferQueue< E
>::transfer (E * e, bool *timed*, long long *nanos*
) throw (decaf::util::concurrent::TimeoutException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]

Performs a put.

Parameters

e the item to be handed to a consumer;

timed if this operation should timeout

nanos the timeout, in nanoseconds

Exceptions

TimeoutException if the operation timed out waiting for the consumer to accept the item offered.

InterruptedException if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3625).

```

6.837.3.2  template<typename E > virtual E*
             decaf::internal::util::concurrent::TransferQueue< E
             >::transfer ( bool timed, long long nanos )
             throw ( decaf::util::concurrent::TimeoutException,
             decaf::lang::exceptions::InterruptedException ) [inline, virtual]

```

Performs a take.

Parameters

timed if this operation should timeout
nanos the timeout, in nanoseconds

Returns

the item provided or received;

Exceptions

TimeoutException if the operation timed out waiting for the producer to offer an item.
InterruptedException if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3625).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**TransferQueue.h**

6.838 decaf::internal::util::concurrent::TransferStack< E > Class Template Reference

```
#include <src/main/decaf/internal/util/concurrent/TransferStack.h>
```

Inheritance diagram for decaf::internal::util::concurrent::TransferStack< E >:

Public Member Functions

- **TransferStack** ()
- virtual ~**TransferStack** ()
- virtual void **transfer** (E *e, bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)
Performs a put.
- virtual E * **transfer** (bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)
Performs a take.


```
template<typename E> class decaf::internal::util::concurrent::TransferStack< E >
```

6.838.1 Constructor & Destructor Documentation

6.838.1.1 `template<typename E > decaf::internal::util::concurrent::TransferStack< E >::TransferStack () [inline]`

6.838.1.2 `template<typename E > virtual decaf::internal::util::concurrent::TransferStack< E >::~~TransferStack () [inline, virtual]`

6.838.2 Member Function Documentation

6.838.2.1 `template<typename E > virtual void decaf::internal::util::concurrent::TransferStack< E >::transfer (E * e, bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Performs a put.

Parameters

e the item to be handed to a consumer;
timed if this operation should timeout
nanos the timeout, in nanoseconds

Exceptions

TimeoutException if the operation timed out waiting for the consumer to accept the item offered.
InterruptedException if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements `decaf::internal::util::concurrent::Transferer< E >` (p. 3625).

6.838.2.2 `template<typename E > virtual E* decaf::internal::util::concurrent::TransferStack< E >::transfer (bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Performs a take.

Parameters

timed if this operation should timeout
nanos the timeout, in nanoseconds

Returns

the item provided or received;

Exceptions

TimeoutException if the operation timed out waiting for the producer to offer an item.

InterruptedException if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3625).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/TransferStack.h`

6.839 activemq::transport::Transport Class Reference

Interface for a transport layer for command objects.

```
#include <src/main/activemq/transport/Transport.h>
```

Inheritance diagram for `activemq::transport::Transport`:

Public Member Functions

- virtual `~Transport()`
- virtual void **start** ()=0 throw (decaf::io::IOException)
*Starts the **Transport** (p. 3629), the send methods of a **Transport** (p. 3629) will throw an exception if used before the **Transport** (p. 3629) is started.*
- virtual void **stop** ()=0 throw (decaf::io::IOException)
*Stops the **Transport** (p. 3629).*
- virtual void **oneway** (const **Pointer**< **Command** > &command)=0 throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)=0 throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout)=0 throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)=0
*Sets the *WireFormat* instance to use.*
- virtual void **setTransportListener** (**TransportListener** *listener)=0
Sets the observer of asynchronous events from this transport.

- virtual **TransportListener** * **getTransportListener** () const =0
Gets the observer of asynchronous events from this transport.
- virtual **Transport** * **narrow** (const std::type_info &typeId)=0
*Narrows down a Chain of Transports to a specific **Transport** (p. 3629) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const =0
*Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const =0
*Is the **Transport** (p. 3629) Connected to its Broker.*
- virtual bool **isClosed** () const =0
*Has the **Transport** (p. 3629) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const =0
- virtual void **reconnect** (const **decaf::net::URI** &uri)=0 throw (**decaf::io::IOException**)
reconnect to another location

6.839.1 Detailed Description

Interface for a transport layer for command objects. Callers can send oneway messages or make synchronous requests. Non-response messages will be delivered to the specified listener object upon receipt. A user of the **Transport** (p. 3629) can set an exception listener to be notified of errors that occurs in Threads that the **Transport** (p. 3629) layer runs. Transports should be given an instance of a WireFormat object when created so that they can turn the built in Commands to / from the required wire format encoding.

6.839.2 Constructor & Destructor Documentation

- 6.839.2.1** virtual **activemq::transport::Transport::~~Transport** () [inline, virtual]

6.839.3 Member Function Documentation

- 6.839.3.1** virtual std::string **activemq::transport::Transport::getRemoteAddress** () const [pure virtual]

Returns

the remote address for this connection

Implemented in **activemq::transport::mock::MockTransport** (p. 2596).

6.839.3.2 `virtual TransportListener* activemq::transport::Transport::getTransportListener () const [pure virtual]`

Gets the observer of asynchronous events from this transport.

Returns

the listener of transport events.

Implemented in `activemq::transport::mock::MockTransport` (p. 2596).

6.839.3.3 `virtual bool activemq::transport::Transport::isClosed () const [pure virtual]`

Has the **Transport** (p. 3629) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3629)

Implemented in `activemq::transport::mock::MockTransport` (p. 2597).

6.839.3.4 `virtual bool activemq::transport::Transport::isConnected () const [pure virtual]`

Is the **Transport** (p. 3629) Connected to its Broker.

Returns

true if a connection has been made.

Implemented in `activemq::transport::mock::MockTransport` (p. 2597).

6.839.3.5 `virtual bool activemq::transport::Transport::isFaultTolerant () const [pure virtual]`

Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3629) is fault tolerant.

Implemented in `activemq::transport::mock::MockTransport` (p. 2597).

6.839.3.6 `virtual Transport* activemq::transport::Transport::narrow (const std::type_info & typeId) [pure virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3629) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

typeId - The type_info of the Object we are searching for.

Returns

the requested Object. or NULL if its not in this chain.

Implemented in **activemq::transport::mock::MockTransport** (p. 2598).

Referenced by **activemq::transport::failover::FailoverTransport::narrow()**.

6.839.3.7 `virtual void activemq::transport::Transport::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implemented in **activemq::transport::mock::MockTransport** (p. 2598).

6.839.3.8 `virtual void activemq::transport::Transport::reconnect (const decaf::net::URI & uri) throw (decaf::io::IOException) [pure virtual]`

reconnect to another location

Parameters

uri

Exceptions

IOException on failure of if not supported

6.839.3.9 `virtual Pointer<Response> activemq::transport::Transport::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

command the command to be sent.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implemented in **activemq::transport::mock::MockTransport** (p. 2599).

```
6.839.3.10  virtual Pointer<Response> activemq::transport::Transport::request
              ( const Pointer< Command > & command, unsigned
                int timeout ) throw ( decaf::io::IOException,
                decaf::lang::exceptions::UnsupportedOperationException ) [pure
                virtual]
```

Sends the given command to the broker and then waits for the response.

Parameters

command - The command to be sent.

timeout - The time to wait for this response.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implemented in **activemq::transport::mock::MockTransport** (p. 2599).

```
6.839.3.11  virtual void activemq::transport::Transport::setTransportListener (
              TransportListener * listener ) [pure virtual]
```

Sets the observer of asynchronous events from this transport.

Parameters

listener the listener of transport events.

Implemented in **activemq::transport::mock::MockTransport** (p. 2601).

```
6.839.3.12  virtual void activemq::transport::Transport::setWireFormat ( const
              Pointer< wireformat::WireFormat > & wireFormat ) [pure virtual]
```

Sets the WireFormat instance to use.

Parameters

wireFormat The WireFormat the object used to encode / decode commands.

6.839.3.13 virtual void activemq::transport::Transport::start () throw (decaf::io::IOException) [pure virtual]

Starts the **Transport** (p. 3629), the send methods of a **Transport** (p. 3629) will throw an exception if used before the **Transport** (p. 3629) is started.

Exceptions

IOException if and error occurs while starting the **Transport** (p. 3629).

Implemented in **activemq::transport::mock::MockTransport** (p. 2601).

6.839.3.14 virtual void activemq::transport::Transport::stop () throw (decaf::io::IOException) [pure virtual]

Stops the **Transport** (p. 3629).

Exceptions

IOException if an error occurs while stopping the transport.

Implemented in **activemq::transport::mock::MockTransport** (p. 2601).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**Transport.h**

6.840 activemq::transport::TransportFactory Class Reference

Defines the interface for Factories that create Transports or TransportFilters.

```
#include <src/main/activemq/transport/TransportFactory.h>
```

Inheritance diagram for activemq::transport::TransportFactory:

Public Member Functions

- virtual ~**TransportFactory** ()
- virtual **Pointer**< **Transport** > **create** (const decaf::net::URI &location)=0 throw (exceptions::ActiveMQException)
*Creates a fully configured **Transport** (p. 3629) instance which could be a chain of filters and transports.*
- virtual **Pointer**< **Transport** > **createComposite** (const decaf::net::URI &location)=0 throw (exceptions::ActiveMQException)
*Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.*

6.840.1 Detailed Description

Defines the interface for Factories that create **Transports** or **TransportFilters**. The factory should be able to create either a completely configured **Transport** (p. 3629) meaning that it has all the appropriate filters wrapping it, or it should be able to create a slimed down version that is used in composite transports like Failover or Fanout.

Since

3.0

6.840.2 Constructor & Destructor Documentation

6.840.2.1 `virtual activemq::transport::TransportFactory::~~TransportFactory ()`
[inline, virtual]

6.840.3 Member Function Documentation

6.840.3.1 `virtual Pointer<Transport> activemq::transport::TransportFactory::create (const decaf::net::URI & location) throw (exceptions::ActiveMQException)`
[pure virtual]

Creates a fully configured **Transport** (p. 3629) instance which could be a chain of filters and transports.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implemented in `activemq::transport::failover::FailoverTransportFactory` (p. 1764), `activemq::transport::mock::MockTransportFactory` (p. 2603), and `activemq::transport::tcp::TcpTransportFactory` (p. 3515).

6.840.3.2 `virtual Pointer<Transport> activemq::transport::TransportFactory::createComposite (const decaf::net::URI & location) throw (exceptions::ActiveMQException)` [pure virtual]

Creates a slimed down **Transport** (p. 3629) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implemented in **activemq::transport::failover::FailoverTransportFactory** (p. 1765), **activemq::transport::mock::MockTransportFactory** (p. 2603), and **activemq::transport::tcp::TcpTransportFactory** (p. 3515).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportFactory.h`

6.841 activemq::transport::TransportFilter Class Reference

A filter on the transport layer.

```
#include <src/main/activemq/transport/TransportFilter.h>
```

Inheritance diagram for `activemq::transport::TransportFilter`:

Public Member Functions

- **TransportFilter** (const **Pointer**< **Transport** > &next)
Constructor.
- virtual **~TransportFilter** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Not supported by this class - throws an exception.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Not supported by this class - throws an exception.
- virtual void **setTransportListener** (**TransportListener** *listener)
Sets the observer of asynchronous exceptions from this transport.

- virtual **TransportListener** * **getTransportListener** () const
Gets the observer of asynchronous exceptions from this transport.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
Sets the WireFormat instance to use.
- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **stop** () throw (decaf::io::IOException)
*Stops the **Transport** (p. 3629).*
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.
- virtual **Transport** * **narrow** (const std::type_info &typeId)
*Narrows down a Chain of Transports to a specific **Transport** (p. 3629) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
*Is the **Transport** (p. 3629) Connected to its Broker.*
- virtual bool **isClosed** () const
*Has the **Transport** (p. 3629) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri) throw (decaf::io::IOException)
reconnect to another location

Protected Member Functions

- void **fire** (const **decaf::lang::Exception** &ex)
Notify the listener of the thrown Exception.
- void **fire** (const **Pointer**< **Command** > &command)
Notify the listener of the new incoming Command.

Protected Attributes

- **Pointer**< **Transport** > **next**
The transport that this filter wraps around.

- **TransportListener * listener**

Listener of this transport.

6.841.1 Detailed Description

A filter on the transport layer. **Transport** (p. 3629) filters implement the **Transport** (p. 3629) interface and optionally delegate calls to another **Transport** (p. 3629) object.

Since

1.0

6.841.2 Constructor & Destructor Documentation

6.841.2.1 **activemq::transport::TransportFilter::TransportFilter** (**const** **Pointer**< **Transport** > & *next*)

Constructor.

Parameters

next - the next **Transport** (p. 3629) in the chain

6.841.2.2 **virtual** **activemq::transport::TransportFilter::~~TransportFilter** () [inline, virtual]

6.841.3 Member Function Documentation

6.841.3.1 **virtual** **void** **activemq::transport::TransportFilter::close** () **throw** (**decaf::io::IOException**) [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if an error occurs while closing the **Transport** (p. 3629).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3082), **activemq::transport::inactivity::InactivityMonitor** (p. 1875), **activemq::transport::tcp::TcpTransport** (p. 3512), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2714).

6.841.3.2 **void** **activemq::transport::TransportFilter::fire** (**const** **Pointer**< **Command** > & *command*) [protected]

Notify the listener of the new incoming Command.

Parameters

command - the command to send to the listener

6.841.3.3 `void activemq::transport::TransportFilter::fire (const decaf::lang::Exception & ex)` [protected]

Notify the listener of the thrown Exception.

Parameters

ex - the exception to send to listeners

6.841.3.4 `virtual std::string activemq::transport::TransportFilter::getRemoteAddress () const` [inline, virtual]

Returns

the remote address for this connection

6.841.3.5 `virtual TransportListener* activemq::transport::TransportFilter::getTransportListener () const` [inline, virtual]

Gets the observer of asynchronous exceptions from this transport.

Returns

The listener of transport events.

6.841.3.6 `virtual bool activemq::transport::TransportFilter::isClosed () const` [inline, virtual]

Has the **Transport** (p. 3629) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3629)

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 3513).

6.841.3.7 `virtual bool activemq::transport::TransportFilter::isConnected () const` [inline, virtual]

Is the **Transport** (p. 3629) Connected to its Broker.

Returns

true if a connection has been made.

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 3513).

6.841.3.8 `virtual bool activemq::transport::TransportFilter::isFaultTolerant ()
const [inline, virtual]`

Is this **Transport** (p. 3629) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3629) is fault tolerant.

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 3513).

6.841.3.9 `virtual Transport* activemq::transport::TransportFilter::narrow (const
std::type_info & typeId) [virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3629) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

typeId - The type_info of the Object we are searching for.

Returns

the requested Object. or NULL if its not in this chain.

6.841.3.10 `virtual void activemq::transport::TransportFilter::onCommand (const
Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

Parameters

command - the received command object.

Implements **activemq::transport::TransportListener** (p. 3644).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3083), **activemq::transport::inactivity::InactivityMonitor** (p. 1876), **activemq::transport::logging::LoggingTransport** (p. 2253), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2715).

6.841.3.11 `virtual void activemq::transport::TransportFilter::oneway (const
Pointer< Command > & command) throw (decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3083), **activemq::transport::inactivity::InactivityMonitor** (p. 1876), **activemq::transport::logging::LoggingTransport** (p. 2253), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2715).

6.841.3.12 `virtual void activemq::transport::TransportFilter::onException (const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters

ex The exception to handle.

Implements **activemq::transport::TransportListener** (p. 3645).

Reimplemented in **activemq::transport::inactivity::InactivityMonitor** (p. 1876).

6.841.3.13 `virtual void activemq::transport::TransportFilter::reconnect (const decaf::net::URI & uri) throw (decaf::io::IOException) [virtual]`

reconnect to another location

Parameters

uri

Exceptions

IOException on failure of if not supported

6.841.3.14 `virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Not supported by this class - throws an exception.

Parameters

command - The command that is sent as a request

timeout - The the time to wait for a response.

Exceptions

IOException

UnsupportedOperationException.

Reimplemented in `activemq::transport::correlator::ResponseCorrelator` (p. 3084), `activemq::transport::logging::LoggingTransport` (p. 2254), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2716).

6.841.3.15 `virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Not supported by this class - throws an exception.

Parameters

command the command that is sent as a request

Exceptions

IOException

UnsupportedOperationException.

Reimplemented in `activemq::transport::correlator::ResponseCorrelator` (p. 3083), `activemq::transport::logging::LoggingTransport` (p. 2254), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2716).

6.841.3.16 `virtual void activemq::transport::TransportFilter::setTransportListener (TransportListener * listener) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

Parameters

listener the listener of transport events.

6.841.3.17 `virtual void activemq::transport::TransportFilter::setWireFormat (const Pointer< wireformat::WireFormat > & wireFormat) [inline, virtual]`

Sets the WireFormat instance to use.

Parameters

wireFormat The WireFormat the object used to encode / decode commands.

6.841.3.18 `virtual void activemq::transport::TransportFilter::start () throw (decaf::io::IOException) [virtual]`

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions

IOException if an error occurs or if this transport has already been closed.

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3084), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2716).

6.841.3.19 **virtual void activemq::transport::TransportFilter::stop () throw (decaf::io::IOException)** [virtual]

Stops the **Transport** (p. 3629).

Exceptions

IOException if an error occurs while stopping the **Transport** (p. 3629).

6.841.3.20 **virtual void activemq::transport::TransportFilter::transportInterrupted ()** [virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 3645).

6.841.3.21 **virtual void activemq::transport::TransportFilter::transportResumed ()** [virtual]

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 3645).

6.841.4 Field Documentation

6.841.4.1 **TransportListener* activemq::transport::TransportFilter::listener** [protected]

Listener of this transport.

6.841.4.2 **Pointer<Transport> activemq::transport::TransportFilter::next** [protected]

The transport that this filter wraps around.

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportFilter.h**

6.842 activemq::transport::TransportListener Class Reference

A listener of asynchronous exceptions from a command transport object.


```
#include <src/main/activemq/transport/TransportListener.h>
```

Inheritance diagram for activemq::transport::TransportListener:

Public Member Functions

- virtual `~TransportListener ()`
- virtual void `onCommand (const Pointer< Command > &command)=0`
Event handler for the receipt of a command.
- virtual void `onException (const decaf::lang::Exception &ex)=0`
Event handler for an exception from a command transport.
- virtual void `transportInterrupted ()=0`
The transport has suffered an interruption from which it hopes to recover.
- virtual void `transportResumed ()=0`
The transport has resumed after an interruption.

6.842.1 Detailed Description

A listener of asynchronous exceptions from a command transport object.

6.842.2 Constructor & Destructor Documentation

6.842.2.1 virtual `activemq::transport::TransportListener::~~TransportListener ()`
[inline, virtual]

6.842.3 Member Function Documentation

6.842.3.1 virtual void `activemq::transport::TransportListener::onCommand (const Pointer< Command > & command)` [pure virtual]

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3629) deletes the command upon receipt.

Parameters

command the received command object.

Implemented in `activemq::core::ActiveMQConnection` (p. 246), `activemq::transport::correlator::ResponseCorrelator` (p. 3083), `activemq::transport::failover::FailoverTransportListener` (p. 1767), `activemq::transport::inactivity::InactivityMonitor` (p. 1876), `activemq::transport::logging::LoggingTransport` (p. 2253), `activemq::transport::TransportFilter` (p. 3640), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2715).

6.842.3.2 virtual void activemq::transport::TransportListener::onException (const decaf::lang::Exception & *ex*) [pure virtual]

Event handler for an exception from a command transport.

Parameters

ex The exception being propagated to this listener to handle.

Implemented in **activemq::core::ActiveMQConnection** (p. 246),
activemq::transport::failover::BackupTransport (p. 691), **ac-**
tivemq::transport::failover::FailoverTransportListener (p. 1767), **ac-**
tivemq::transport::inactivity::InactivityMonitor (p. 1876), and **ac-**
tivemq::transport::TransportFilter (p. 3641).

6.842.3.3 virtual void activemq::transport::TransportListener::transportInterrupted () [pure virtual]

The transport has suffered an interruption from which it hopes to recover.

Implemented in **activemq::core::ActiveMQConnection** (p. 251),
activemq::transport::DefaultTransportListener (p. 1594), **ac-**
tivemq::transport::failover::FailoverTransportListener (p. 1767), and **ac-**
tivemq::transport::TransportFilter (p. 3643).

6.842.3.4 virtual void activemq::transport::TransportListener::transportResumed () [pure virtual]

The transport has resumed after an interruption.

Implemented in **activemq::core::ActiveMQConnection** (p. 251),
activemq::transport::DefaultTransportListener (p. 1594), **ac-**
tivemq::transport::failover::FailoverTransportListener (p. 1767), and **ac-**
tivemq::transport::TransportFilter (p. 3643).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportListener.h`

6.843 activemq::transport::TransportRegistry Class Reference

Registry of all **Transport** (p. 3629) Factories that are available to the client at runtime.

```
#include <src/main/activemq/transport/TransportRegistry.h>
```

Public Member Functions

- virtual **~TransportRegistry** ()
- **TransportFactory** * **findFactory** (const std::string &name) const throw (decaf::lang::exceptions::NoSuchElementException)

*Gets a Registered **TransportFactory** (p. 3634) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*

- void **registerFactory** (const std::string &name, **TransportFactory** *factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

*Registers a new **TransportFactory** (p. 3634) with this Registry.*

- void **unregisterFactory** (const std::string &name)

Unregisters the Factory with the given name and deletes that instance of the Factory.

- std::vector< std::string > **getTransportNames** () const

Retrieves a list of the names of all the Registered Transport's in this Registry.

Static Public Member Functions

- static **TransportRegistry** & **getInstance** ()

*Gets the single instance of the **TransportRegistry** (p. 3645).*

6.843.1 Detailed Description

Registry of all **Transport** (p. 3629) Factories that are available to the client at runtime. New Transport's must have a factory registered here before a connection attempt is made.

Since

3.0

6.843.2 Constructor & Destructor Documentation

- 6.843.2.1 virtual **activemq::transport::TransportRegistry::~TransportRegistry** ()
[virtual]

6.843.3 Member Function Documentation

- 6.843.3.1 **TransportFactory*** **activemq::transport::TransportRegistry::findFactory**
(const std::string & *name*) const throw (decaf::lang::exceptions::NoSuchElementException)

Gets a Registered **TransportFactory** (p. 3634) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

Parameters

name The name of the Factory to find in the Registry.

Returns

the Factory registered under the given name.

Exceptions

NoSuchElementException if no factory is registered with that name.

6.843.3.2 `static TransportRegistry& activemq::transport::TransportRegistry::getInstance ()`
[static]

Gets the single instance of the **TransportRegistry** (p. 3645).

Returns

reference to the single instance of this Registry

6.843.3.3 `std::vector<std::string> activemq::transport::TransportRegistry::getTransportNames () const`

Retrieves a list of the names of all the Registered Transport's in this Registry.

Returns

stl vector of strings with all the **Transport** (p. 3629) names registered.

6.843.3.4 `void activemq::transport::TransportRegistry::registerFactory (const std::string & name, TransportFactory * factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)`

Registers a new **TransportFactory** (p. 3634) with this Registry.

If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

Parameters

name The name of the new Factory to register.

factory The new Factory to add to the Registry.

Exceptions

IllegalArgumentException is name is the empty string.

NullPointerException if the Factory is Null.

6.843.3.5 `void activemq::transport::TransportRegistry::unregisterFactory (const std::string & name)`

Unregisters the Factory with the given name and deletes that instance of the Factory.

Parameters

- name* Name of the Factory to unregister and destroy

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportRegistry.h**

6.844 tree_desc_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

Data Fields

- ct_data * dyn_tree
- int max_code
- static_tree_desc * stat_desc

6.844.1 Field Documentation

6.844.1.1 ct_data* tree_desc_s::dyn_tree

6.844.1.2 int tree_desc_s::max_code

6.844.1.3 static_tree_desc* tree_desc_s::stat_desc

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**deflate.h**

6.845 decaf::lang::Thread::UncaughtExceptionHandler Class Reference

Interface for handlers invoked when a **Thread** (p. 3520) abruptly terminates due to an uncaught exception.

```
#include <src/main/decaf/lang/Thread.h>
```

Public Member Functions

- virtual ~**UncaughtExceptionHandler** ()
- virtual void **uncaughtException** (const **Thread** *thread, const **Throwable** &error)=0
throw ()

Method invoked when the given thread terminates due to the given uncaught exception.

6.845.1 Detailed Description

Interface for handlers invoked when a **Thread** (p. 3520) abruptly terminates due to an uncaught exception.

6.845.2 Constructor & Destructor Documentation

6.845.2.1 **virtual**
decaf::lang::Thread::UncaughtExceptionHandler::~~UncaughtExceptionHandler
() [inline, virtual]

6.845.3 Member Function Documentation

6.845.3.1 **virtual void de-**
caf::lang::Thread::UncaughtExceptionHandler::uncaughtException (
const Thread * *thread*, const Throwable & *error*) throw () [pure
 virtual]

Method invoked when the given thread terminates due to the given uncaught exception.

This method is defined to indicate that it will not throw an exception, throwing an exception from this method will on most systems result in a segmentation fault.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Thread.h**

6.846 decaf::net::UnknownHostException Class Reference

```
#include <src/main/decaf/net/UnknownHostException.h>
```

Inheritance diagram for decaf::net::UnknownHostException:

Public Member Functions

- **UnknownHostException ()** throw ()
Default Constructor.
- **UnknownHostException (const Exception &ex)** throw ()
Conversion Constructor from some other Exception.
- **UnknownHostException (const UnknownHostException &ex)** throw ()
Copy Constructor.
- **UnknownHostException (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...)** throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnknownHostException (const std::exception *cause)** throw ()

Constructor.

- **UnknownHostException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **UnknownHostException** * clone () const

Clones this exception.

- virtual ~**UnknownHostException** () throw ()

6.846.1 Constructor & Destructor Documentation

6.846.1.1 decaf::net::UnknownHostException::UnknownHostException () throw () [inline]

Default Constructor.

6.846.1.2 decaf::net::UnknownHostException::UnknownHostException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.846.1.3 decaf::net::UnknownHostException::UnknownHostException (const UnknownHostException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.846.1.4 decaf::net::UnknownHostException::UnknownHostException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.846.1.5 `decaf::net::UnknownHostException::UnknownHostException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.846.1.6 `decaf::net::UnknownHostException::UnknownHostException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.846.1.7 `virtual decaf::net::UnknownHostException::~~UnknownHostException () throw () [inline, virtual]`

6.846.2 Member Function Documentation

6.846.2.1 `virtual UnknownHostException* decaf::net::UnknownHostException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::IOException` (p. 2005).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/UnknownHostException.h`

6.847 decaf::net::UnknownServiceException Class Reference

```
#include <src/main/decaf/net/UnknownServiceException.h>
```

Inheritance diagram for decaf::net::UnknownServiceException:

Public Member Functions

- **UnknownServiceException** () throw ()
Default Constructor.
- **UnknownServiceException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **UnknownServiceException** (const **UnknownServiceException** &ex) throw ()
Copy Constructor.
- **UnknownServiceException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnknownServiceException** (const std::exception *cause) throw ()
Constructor.
- **UnknownServiceException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnknownServiceException** * clone () const
Clones this exception.
- virtual ~**UnknownServiceException** () throw ()

6.847.1 Constructor & Destructor Documentation

6.847.1.1 decaf::net::UnknownServiceException::UnknownServiceException () throw () [inline]

Default Constructor.

6.847.1.2 decaf::net::UnknownServiceException::UnknownServiceException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.847.1.3 `decaf::net::UnknownServiceException::UnknownServiceException (const UnknownServiceException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.847.1.4 `decaf::net::UnknownServiceException::UnknownServiceException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.847.1.5 `decaf::net::UnknownServiceException::UnknownServiceException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.847.1.6 `decaf::net::UnknownServiceException::UnknownServiceException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.847.1.7 `virtual`
`decaf::net::UnknownServiceException::~UnknownServiceException ()`
`throw ()` [`inline`, `virtual`]

6.847.2 Member Function Documentation

6.847.2.1 `virtual UnknownServiceException* decaf::net::UnknownServiceException::clone () const`
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::IOException` (p. 2005).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/UnknownServiceException.h`

6.848 decaf::io::UnsupportedEncodingException Class Reference

Thrown when the the Character Encoding is not supported.

```
#include <src/main/decaf/io/UnsupportedEncodingException.h>
```

Inheritance diagram for `decaf::io::UnsupportedEncodingException`:

Public Member Functions

- `UnsupportedEncodingException () throw ()`
Default Constructor.
- `UnsupportedEncodingException (const lang::Exception &ex) throw ()`
Copy Constructor.
- `UnsupportedEncodingException (const UnsupportedEncodingException &ex) throw ()`
Copy Constructor.
- `UnsupportedEncodingException (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()`
Constructor - Initializes the file name and line number where this message occurred.

- **UnsupportedEncodingException** (const std::exception ***cause**) throw ()
Constructor.
- **UnsupportedEncodingException** (const char ***file**, const int **lineNumber**, const char ***msg**,...) throw ()
Constructor.
- virtual **UnsupportedEncodingException * clone** () const
Clones this exception.
- virtual ~**UnsupportedEncodingException** () throw ()

6.848.1 Detailed Description

Thrown when the the Character Encoding is not supported.

Since

1.0

6.848.2 Constructor & Destructor Documentation

6.848.2.1 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException () throw () [inline]

Default Constructor.

6.848.2.2 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.848.2.3 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const UnsupportedEncodingException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.848.2.4 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.848.2.5 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException
(const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.848.2.6 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException
(const char * file, const int lineNumber, const char * msg, ...)
throw () [inline]`

Constructor.

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.848.2.7 `virtual
decaf::io::UnsupportedEncodingException::~~UnsupportedEncodingException
() throw () [inline, virtual]`

6.848.3 Member Function Documentation

6.848.3.1 `virtual UnsupportedEncodingException* de-
caf::io::UnsupportedEncodingException::clone () const [inline,
virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 2005).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/UnsupportedEncodingException.h`

6.849 decaf::lang::exceptions::UnsupportedOperationException Class Reference

```
#include <src/main/decaf/lang/exceptions/UnsupportedOperationException.h>
```

Inheritance diagram for `decaf::lang::exceptions::UnsupportedOperationException`:

Public Member Functions

- **UnsupportedOperationException** () throw ()
Default Constructor.
- **UnsupportedOperationException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1712).*
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()
Copy Constructor.
- **UnsupportedOperationException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnsupportedOperationException** (const std::exception *cause) throw ()
Constructor.
- **UnsupportedOperationException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnsupportedOperationException** * clone () const
Clones this exception.
- virtual ~**UnsupportedOperationException** () throw ()

6.849.1 Constructor & Destructor Documentation

6.849.1.1 **decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException** () throw () [inline]

Default Constructor.

6.849.1.2 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1712).

Parameters

ex An exception that should become this type of **Exception** (p. 1712)

6.849.1.3 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const UnsupportedOperationException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of **Exception** (p. 1712)

6.849.1.4 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const char * *file*, const int *lineNumber*, const std::exception *
cause, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.849.1.5 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause **Pointer** (p. 2756) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.849.1.6 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const char * *file*, const int *lineNumber*, const char * *msg*, ...)
throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.849.1.7 **virtual**
decaf::lang::exceptions::UnsupportedOperationException::~~UnsupportedOperationException
() throw () [inline, virtual]

6.849.2 Member Function Documentation

6.849.2.1 **virtual UnsupportedOperationException* de-**
caf::lang::exceptions::UnsupportedOperationException::clone () const
 [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

- an new **Exception** (p. 1712) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1715).

Reimplemented in **decaf::nio::ReadOnlyBufferException** (p. 2968).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/UnsupportedOperationException.h`

6.850 cms::UnsupportedOperationException Class Reference

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

```
#include <src/main/cms/UnsupportedOperationException.h>
```

Inheritance diagram for cms::UnsupportedOperationException:

Public Member Functions

- **UnsupportedOperationException** () throw ()
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()

- **UnsupportedOperationException** (const std::string &message, const std::exception *cause) throw ()
- **UnsupportedOperationException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**UnsupportedOperationException** () throw ()

6.850.1 Detailed Description

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

Since

2.0

6.850.2 Constructor & Destructor Documentation

6.850.2.1 `cms::UnsupportedOperationException::UnsupportedOperationException () throw ()`

6.850.2.2 `cms::UnsupportedOperationException::UnsupportedOperationException (const UnsupportedOperationException & ex) throw ()`

6.850.2.3 `cms::UnsupportedOperationException::UnsupportedOperationException (const std::string & message, const std::exception * cause) throw ()`

6.850.2.4 `cms::UnsupportedOperationException::UnsupportedOperationException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`

6.850.2.5 `virtual cms::UnsupportedOperationException::~~UnsupportedOperationException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/UnsupportedOperationException.h`

6.851 decaf::net::URI Class Reference

This class represents an instance of a **URI** (p. 3660) as defined by RFC 2396.

```
#include <src/main/decaf/net/URI.h>
```

Inheritance diagram for decaf::net::URI:

Public Member Functions

- **URI** ()

Default Constructor, same as calling a Constructor with all fields empty.

- **URI** (const **URI** &uri) throw (**URISyntaxException**)
*Constructs a **URI** (p. 3660) as a copy of another **URI** (p. 3660).*
- **URI** (const std::string &uri) throw (**URISyntaxException**)
*Constructs a **URI** (p. 3660) from the given string.*
- **URI** (const std::string &scheme, const std::string &ssp, const std::string &fragment) throw (**URISyntaxException**)
*Constructs a **URI** (p. 3660) from the given components.*
- **URI** (const std::string &scheme, const std::string &userInfo, const std::string &host, int port, const std::string &path, const std::string &query, const std::string &fragment) throw (**URISyntaxException**)
*Constructs a **URI** (p. 3660) from the given components.*
- **URI** (const std::string &scheme, const std::string &host, const std::string &path, const std::string &fragment) throw (**URISyntaxException**)
*Constructs a **URI** (p. 3660) from the given components.*
- **URI** (const std::string &scheme, const std::string &authority, const std::string &path, const std::string &query, const std::string &fragment) throw (**URISyntaxException**)
*Constructs a **URI** (p. 3660) from the given components.*
- virtual ~**URI** ()
- virtual int **compareTo** (const **URI** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **URI** &value) const
- virtual bool **operator==** (const **URI** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **URI** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **getAuthority** () const
- std::string **getFragment** () const
- std::string **getHost** () const
- std::string **getPath** () const
- int **getPort** () const
- std::string **getQuery** () const
- std::string **getScheme** () const
- std::string **getUserInfo** () const
- std::string **getRawAuthority** () const
*Returns the raw authority component of this **URI** (p. 3660).*
- std::string **getRawFragment** () const
*Returns the raw fragment component of this **URI** (p. 3660).*

- `std::string getRawPath () const`
Returns the raw path component of this **URI** (p. 3660).
- `std::string getRawQuery () const`
Returns the raw query component of this **URI** (p. 3660).
- `std::string getRawSchemeSpecificPart () const`
Returns the raw scheme-specific part of this **URI** (p. 3660).
- `std::string getSchemeSpecificPart () const`
Returns the decoded scheme-specific part of this **URI** (p. 3660).
- `std::string getRawUserInfo () const`
Returns the raw user-information component of this **URI** (p. 3660).
- `bool isAbsolute () const`
Tells whether or not this **URI** (p. 3660) is absolute.
- `bool isOpaque () const`
Tells whether or not this **URI** (p. 3660) is opaque.
- `URI normalize () const`
Normalizes this **URI**'s path.
- `URI parseServerAuthority () const throw (URISyntaxException)`
Attempts to parse this **URI**'s authority component, if defined, into user-information, host, and port components.
- `URI relativize (const URI &uri) const`
Relativizes the given **URI** (p. 3660) against this **URI** (p. 3660).
- `URI resolve (const std::string &str) const throw (lang::exceptions::IllegalArgumentException)`
Constructs a new **URI** (p. 3660) by parsing the given string and then resolving it against this **URI** (p. 3660).
- `URI resolve (const URI &uri) const`
Resolves the given **URI** (p. 3660) against this **URI** (p. 3660).
- `std::string toString () const`
Returns the content of this **URI** (p. 3660) as a string.
- `URL toURL () const throw (MalformedURLException, lang::exceptions::IllegalArgumentException)`
Constructs a **URL** (p. 3697) from this **URI** (p. 3660).

Static Public Member Functions

- static **URI** **create** (const std::string uri) throw (lang::exceptions::IllegalArgumentException)

*Creates a **URI** (p. 3660) by parsing the given string.*

6.851.1 Detailed Description

This class represents an instance of a **URI** (p. 3660) as defined by RFC 2396.

6.851.2 Constructor & Destructor Documentation

6.851.2.1 decaf::net::URI::URI ()

Default Constructor, same as calling a Constructor with all fields empty.

6.851.2.2 decaf::net::URI::URI (const **URI** & *uri*) throw (URISyntaxException)

Constructs a **URI** (p. 3660) as a copy of another **URI** (p. 3660).

Parameters

uri - uri to copy

6.851.2.3 decaf::net::URI::URI (const std::string & *uri*) throw (URISyntaxException)

Constructs a **URI** (p. 3660) from the given string.

Parameters

uri - string uri to parse.

6.851.2.4 decaf::net::URI::URI (const std::string & *scheme*, const std::string & *ssp*, const std::string & *fragment*) throw (URISyntaxException)

Constructs a **URI** (p. 3660) from the given components.

Parameters

scheme - the uri scheme

ssp - Scheme specific part

fragment - Fragment

6.851.2.5 decaf::net::URI::URI (const std::string & *scheme*, const std::string & *userInfo*, const std::string & *host*, int *port*, const std::string & *path*, const std::string & *query*, const std::string & *fragment*) throw (URISyntaxException)

Constructs a **URI** (p. 3660) from the given components.

Parameters

scheme - Scheme name

userInfo - User name and authorization information

host - Host name

port - Port number

path - Path

query - Query

fragment - Fragment

6.851.2.6 decaf::net::URI::URI (const std::string & *scheme*, const std::string & *host*, const std::string & *path*, const std::string & *fragment*) throw (URISyntaxException)

Constructs a **URI** (p. 3660) from the given components.

Parameters

scheme - Scheme name

host - Host name

path - Path

fragment - Fragment

6.851.2.7 decaf::net::URI::URI (const std::string & *scheme*, const std::string & *authority*, const std::string & *path*, const std::string & *query*, const std::string & *fragment*) throw (URISyntaxException)

Constructs a **URI** (p. 3660) from the given components.

Parameters

scheme - Scheme name

authority - Authority

path - Path

query - Query

fragment - Fragment

6.851.2.8 `virtual decaf::net::URI::~~URI () [inline, virtual]`

6.851.3 Member Function Documentation

6.851.3.1 `virtual int decaf::net::URI::compareTo (const URI & value) const [virtual]`

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters

value - the value to compare to this one.

Returns

zero if equal minus one if less than and one if greater than.

6.851.3.2 `static URI decaf::net::URI::create (const std::string uri) throw (lang::exceptions::IllegalArgumentException) [static]`

Creates a **URI** (p. 3660) by parsing the given string.

This convenience factory method works as if by invoking the `URI(string)` constructor; any **URISyntaxException** (p. 3686) thrown by the constructor is caught and wrapped in a new `IllegalArgumentException` object, which is then thrown.

Parameters

uri - **URI** (p. 3660) string to parse

Exceptions

IllegalArgumentException

6.851.3.3 `virtual bool decaf::net::URI::equals (const URI & value) const [virtual]`

Returns

true if this value is considered equal to the passed value.

6.851.3.4 `std::string decaf::net::URI::getAuthority () const`

Returns

the decoded authority component of this **URI** (p. 3660).

6.851.3.5 `std::string decaf::net::URI::getFragment () const`

Returns

the decoded fragment component of this **URI** (p. 3660).

6.851.3.6 `std::string decaf::net::URI::getHost () const`**Returns**

the host component of this **URI** (p. 3660).

6.851.3.7 `std::string decaf::net::URI::getPath () const`**Returns**

the path component of this **URI** (p. 3660).

6.851.3.8 `int decaf::net::URI::getPort () const`**Returns**

the port component of this **URI** (p. 3660).

6.851.3.9 `std::string decaf::net::URI::getQuery () const`**Returns**

the query component of this **URI** (p. 3660).

6.851.3.10 `std::string decaf::net::URI::getRawAuthority () const`

Returns the raw authority component of this **URI** (p. 3660).

The authority component of a **URI** (p. 3660), if defined, only contains the commercial-at character ('@') and characters in the unreserved, punct, escaped, and other categories. If the authority is server-based then it is further constrained to have valid user-information, host, and port components.

Returns

the raw authority component of the **URI** (p. 3660)

6.851.3.11 `std::string decaf::net::URI::getRawFragment () const`

Returns the raw fragment component of this **URI** (p. 3660).

The fragment component of a **URI** (p. 3660), if defined, only contains legal **URI** (p. 3660) characters.

Returns

the raw fragment component of this **URI** (p. 3660)

6.851.3.12 `std::string decaf::net::URI::getRawPath () const`

Returns the raw path component of this **URI** (p. 3660).

The path component of a **URI** (p. 3660), if defined, only contains the slash character ('/'), the commercial-at character ('@'), and characters in the unreserved, punct, escaped, and other categories.

Returns

the raw path component of this **URI** (p. 3660)

6.851.3.13 `std::string decaf::net::URI::getRawQuery () const`

Returns the raw query component of this **URI** (p. 3660).

The query component of a **URI** (p. 3660), if defined, only contains legal **URI** (p. 3660) characters.

Returns

the raw query component of the **URI** (p. 3660).

6.851.3.14 `std::string decaf::net::URI::getRawSchemeSpecificPart () const`

Returns the raw scheme-specific part of this **URI** (p. 3660).

The scheme-specific part is never undefined, though it may be empty. The scheme-specific part of a **URI** (p. 3660) only contains legal **URI** (p. 3660) characters.

Returns

the raw scheme special part of the uri

6.851.3.15 `std::string decaf::net::URI::getRawUserInfo () const`

Returns the raw user-information component of this **URI** (p. 3660).

The user-information component of a **URI** (p. 3660), if defined, only contains characters in the unreserved, punct, escaped, and other categories.

Returns

the raw user-information component of the **URI** (p. 3660)

6.851.3.16 `std::string decaf::net::URI::getScheme () const`**Returns**

the scheme component of this **URI** (p. 3660)

6.851.3.17 `std::string decaf::net::URI::getSchemeSpecificPart () const`

Returns the decoded scheme-specific part of this **URI** (p. 3660).

The string returned by this method is equal to that returned by the `getRawSchemeSpecificPart` method except that all sequences of escaped octets are decoded.

Returns

the raw scheme specific part of the uri.

6.851.3.18 `std::string decaf::net::URI::getUserInfo () const`**Returns**

the user info component of this **URI** (p. 3660)

6.851.3.19 `bool decaf::net::URI::isAbsolute () const`

Tells whether or not this **URI** (p. 3660) is absolute.

A **URI** (p. 3660) is absolute if, and only if, it has a scheme component.

Returns

true if, and only if, this **URI** (p. 3660) is absolute

6.851.3.20 `bool decaf::net::URI::isOpaque () const`

Tells whether or not this **URI** (p. 3660) is opaque.

A **URI** (p. 3660) is opaque if, and only if, it is absolute and its scheme-specific part does not begin with a slash character ('/'). An opaque **URI** (p. 3660) has a scheme, a scheme-specific part, and possibly a fragment; all other components are undefined.

Returns

true if, and only if, this **URI** (p. 3660) is opaque

6.851.3.21 `URI decaf::net::URI::normalize () const`

Normalizes this **URI**'s path.

If this **URI** (p. 3660) is opaque, or if its path is already in normal form, then this **URI** (p. 3660) is returned. Otherwise a new **URI** (p. 3660) is constructed that is identical to this **URI** (p. 3660) except that its path is computed by normalizing this **URI**'s path in a manner consistent with RFC 2396, section 5.2, step 6, sub-steps c through f; that is:

1. All "." segments are removed.
2. If a ".." segment is preceded by a non- ".." segment then both of these segments are removed. This step is repeated until it is no longer applicable.
3. If the path is relative, and if its first segment contains a colon character (':'), then a "." segment is prepended. This prevents a relative **URI** (p. 3660) with a path such as "a:b/c/d" from later being re-parsed as an opaque **URI** (p. 3660) with a scheme of "a" and a scheme-specific part of "b/c/d". (Deviation from RFC 2396)

A normalized path will begin with one or more "." segments if there were insufficient non- "." segments preceding them to allow their removal. A normalized path will begin with a "." segment if one was inserted by step 3 above. Otherwise, a normalized path will not contain any "." or "." segments.

Returns

A **URI** (p. 3660) equivalent to this **URI** (p. 3660), but whose path is in normal form

6.851.3.22 `virtual bool decaf::net::URI::operator< (const URI & value) const`
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.851.3.23 `virtual bool decaf::net::URI::operator== (const URI & value) const`
[virtual]

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.851.3.24 `URI decaf::net::URI::parseServerAuthority () const throw (`
`URISyntaxException)`

Attempts to parse this URI's authority component, if defined, into user-information, host, and port components.

If this URI's authority component has already been recognized as being server-based then it will already have been parsed into user-information, host, and port components. In this case, or if this **URI** (p. 3660) has no authority component, this method simply returns this **URI** (p. 3660).

Otherwise this method attempts once more to parse the authority component into user-information, host, and port components, and throws an exception describing why the authority component could not be parsed in that way.

Returns

A **URI** (p. 3660) whose authority field has been parsed as a server-based authority

Exceptions

URISyntaxException (p. 3686) - If the authority component of this **URI** (p. 3660) is defined but cannot be parsed as a server-based authority.

6.851.3.25 URI decaf::net::URI::relativize (const URI & uri) const

Relativizes the given **URI** (p. 3660) against this **URI** (p. 3660).

The relativization of the given **URI** (p. 3660) against this **URI** (p. 3660) is computed as follows:

1. If either this **URI** (p. 3660) or the given **URI** (p. 3660) are opaque, or if the scheme and authority components of the two URIs are not identical, or if the path of this **URI** (p. 3660) is not a prefix of the path of the given **URI** (p. 3660), then the given **URI** (p. 3660) is returned.
2. Otherwise a new relative hierarchical **URI** (p. 3660) is constructed with query and fragment components taken from the given **URI** (p. 3660) and with a path component computed by removing this URI's path from the beginning of the given URI's path.

Parameters

uri - The **URI** (p. 3660) to be relativized against this **URI** (p. 3660)

Returns

The resulting **URI** (p. 3660)

6.851.3.26 URI decaf::net::URI::resolve (const std::string & str) const throw (lang::exceptions::IllegalArgumentException)

Constructs a new **URI** (p. 3660) by parsing the given string and then resolving it against this **URI** (p. 3660).

This convenience method works as if invoking it were equivalent to evaluating the expression `resolve(URI::create(str))`.

Parameters

str - The string to be parsed into a **URI** (p. 3660)

Returns

The resulting **URI** (p. 3660)

Exceptions

IllegalArgumentException - If the given string violates RFC 2396

6.851.3.27 URI decaf::net::URI::resolve (const URI & uri) const

Resolves the given **URI** (p. 3660) against this **URI** (p. 3660).

If the given **URI** (p. 3660) is already absolute, or if this **URI** (p. 3660) is opaque, then a copy of the given **URI** (p. 3660) is returned.

If the given URI's fragment component is defined, its path component is empty, and its scheme, authority, and query components are undefined, then a **URI** (p. 3660) with the given fragment but with all other components equal to those of this **URI** (p. 3660) is returned. This allows a **URI** (p. 3660) representing a standalone fragment reference, such as "#foo", to be usefully resolved against a base **URI** (p. 3660).

Otherwise this method constructs a new hierarchical **URI** (p. 3660) in a manner consistent with RFC 2396, section 5.2; that is:

1. A new **URI** (p. 3660) is constructed with this URI's scheme and the given URI's query and fragment components.
2. If the given **URI** (p. 3660) has an authority component then the new URI's authority and path are taken from the given **URI** (p. 3660).
3. Otherwise the new URI's authority component is copied from this **URI** (p. 3660), and its path is computed as follows:

1. If the given URI's path is absolute then the new URI's path is taken from the given **URI** (p. 3660).
2. Otherwise the given URI's path is relative, and so the new URI's path is computed by resolving the path of the given **URI** (p. 3660) against the path of this **URI** (p. 3660). This is done by concatenating all but the last segment of this URI's path, if any, with the given URI's path and then normalizing the result as if by invoking the `normalize` method.

The result of this method is absolute if, and only if, either this **URI** (p. 3660) is absolute or the given **URI** (p. 3660) is absolute.

Parameters

uri - The **URI** (p. 3660) to be resolved against this **URI** (p. 3660)

Returns

The resulting **URI** (p. 3660)

6.851.3.28 `std::string decaf::net::URI::toString () const`

Returns the content of this **URI** (p. 3660) as a string.

If this **URI** (p. 3660) was created by invoking one of the constructors in this class then a string equivalent to the original input string, or to the string computed from the originally-given components, as appropriate, is returned. Otherwise this **URI** (p. 3660) was created by normalization, resolution, or relativization, and so a string is constructed from this URI's components according to the rules specified in RFC 2396, section 5.2, step 7.

Returns

the string form of this **URI** (p. 3660)

6.851.3.29 `URL decaf::net::URI::toURL () const throw (MalformedURLException, lang::exceptions::IllegalArgumentException)`

Constructs a **URL** (p. 3697) from this **URI** (p. 3660).

This convenience method works as if invoking it were equivalent to evaluating the expression `new URL (p. 3697)(this.toString())` after first checking that this **URI** (p. 3660) is absolute.

Returns

A **URL** (p. 3697) constructed from this **URI** (p. 3660)

Exceptions

IllegalArgumentException - If this **URL** (p. 3697) is not absolute

MalformedURLException (p. 2303) - If a protocol handler for the **URL** (p. 3697) could not be found, or if some other error occurred while constructing the **URL** (p. 3697)

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URI.h**

6.852 decaf::internal::net::URIEncoderDecoder Class Reference

```
#include <src/main/decaf/internal/net/URIEncoderDecoder.h>
```

Public Member Functions

- **URIEncoderDecoder** ()
- virtual **~URIEncoderDecoder** ()

Static Public Member Functions

- static void **validate** (const std::string &s, const std::string &legal) throw (decaf::net::URISyntaxException)

Validate a string by checking if it contains any characters other than:

- static void **validateSimple** (const std::string &s, const std::string &legal) throw (decaf::net::URISyntaxException)

Validate a string by checking if it contains any characters other than:

- static std::string **quoteIllegal** (const std::string &s, const std::string &legal)

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by ".

- static std::string **encodeOthers** (const std::string &s)

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.

- static std::string **decode** (const std::string &s)

*Decodes the string argument which is assumed to be encoded in the **x-www-form-urlencoded** MIME content type using the UTF-8 encoding scheme.*

6.852.1 Constructor & Destructor Documentation

6.852.1.1 `decaf::internal::net::URIEncoderDecoder::URIEncoderDecoder ()`

6.852.1.2 `virtual decaf::internal::net::URIEncoderDecoder::~~URIEncoderDecoder () [inline, virtual]`

6.852.2 Member Function Documentation

6.852.2.1 `static std::string decaf::internal::net::URIEncoderDecoder::decode (const std::string & s) [static]`

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme.

” and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A%20B%20C %24%25" -> "A B C \$%"

Parameters

s - The encoded string.

Returns

The decoded version.

6.852.2.2 `static std::string decaf::internal::net::URIEncoderDecoder::encodeOthers (const std::string & s) [static]`

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.

They are converted into their hexadecimal value prepended by ”.

For example: Euro currency symbol -> "%E2%82%AC".

Parameters

s - the string to be converted

Returns

the converted string

6.852.2.3 `static std::string decaf::internal::net::URIEncoderDecoder::quoteIllegal (const std::string & s, const std::string & legal) [static]`

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by ”.

For example: '#' -> "23"

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars, are preserved.

Parameters

- s* - the string to be converted
legal - the characters allowed to be preserved in the string *s*

Returns

converted string

6.852.2.4 static void decaf::internal::net::URIEncoderDecoder::validate
(const std::string & *s*, const std::string & *legal*) throw (
decaf::net::URISyntaxException) [static]

Validate a string by checking if it contains any characters other than:

1. letters ('a'..'z', 'A'..'Z')
2. numbers ('0'..'9')
3. characters in the legalset parameter
4. characters that are not ISO Control or are not ISO Space characters)

Parameters

- s* - the string to be validated
legal - the characters allowed in the string *s*

6.852.2.5 static void decaf::internal::net::URIEncoderDecoder::validateSimple
(const std::string & *s*, const std::string & *legal*) throw (
decaf::net::URISyntaxException) [static]

Validate a string by checking if it contains any characters other than:

1. letters ('a'..'z', 'A'..'Z')
2. numbers ('0'..'9')
3. characters in the legalset parameter

Parameters

- s* - the string to be validated
legal - the characters allowed in the string *s*

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/URIEncoderDecoder.h

6.853 decaf::internal::net::URIHelper Class Reference

Helper class used by the URI classes in encoding and decoding of URI's.

```
#include <src/main/decaf/internal/net/URIHelper.h>
```

Public Member Functions

- **URIHelper** (const std::string &unreserved, const std::string &punct, const std::string &reserved, const std::string &someLegal, const std::string &allLegal)

Setup the **URIHelper** (p. 3674) with values assigned to the various fields that are used in the validation process.

- **URIHelper** ()

Sets up the filter strings with sane defaults.

- virtual **~URIHelper** ()

- **URIType parseURI** (const std::string &uri, bool forceServer) throw (decaf::net::URISyntaxException)

Parse the passed in URI.

- void **validateScheme** (const std::string &uri, const std::string &scheme, int index) throw (decaf::net::URISyntaxException)

Validate the schema portin of the URI.

- void **validateSsp** (const std::string &uri, const std::string &ssp, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI Ssp Segment contains no invalid encodings.

- void **validateAuthority** (const std::string &uri, const std::string &authority, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI Authority Segment contains no invalid encodings.

- void **validatePath** (const std::string &uri, const std::string &path, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI Path Segment contains no invalid encodings.

- void **validateQuery** (const std::string &uri, const std::string &query, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI Query Segment contains no invalid encodings.

- void **validateFragment** (const std::string &uri, const std::string &fragment, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI fragment contains no invalid encodings.

- **URIType parseAuthority** (bool forceServer, const std::string &authority) throw (decaf::net::URISyntaxException)

determine the host, port and user-info if the authority parses successfully to a server based authority

- void **validateUserinfo** (const std::string &uri, const std::string &userinfo, std::size_t index) throw (decaf::net::URISyntaxException)

Check the supplied user info for validity.

- bool **isValidHost** (bool forceServer, const std::string &host) throw (decaf::net::URISyntaxException)

distinguish between IPv4, IPv6, domain name and validate it based on its type

- bool **isValidDomainName** (const std::string &host)

Validates the string past to determine if it is a well formed domain name.

- **bool isValidIPv4Address** (const std::string &host)
Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9.
- **bool isValidIPv6Address** (const std::string &ipAddress)
Determines if the given address is valid according to the IPv6 spec.
- **bool isValidIPv4Word** (const std::string &word)
Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.
- **bool isValidHexChar** (char c)
Determines if the given char is a valid Hex char.

6.853.1 Detailed Description

Helper class used by the URI classes in encoding and decoding of URI's.

6.853.2 Constructor & Destructor Documentation

6.853.2.1 decaf::internal::net::URIHelper::URIHelper (const std::string & *unreserved*, const std::string & *punct*, const std::string & *reserved*, const std::string & *someLegal*, const std::string & *allLegal*)

Setup the **URIHelper** (p. 3674) with values assigned to the various fields that are used in the validation process.

The defaults are overridden by these values.

Parameters

unreserved - characters not reserved for use.
punct - allowable punctuation symbols.
reserved - characters not allowed for general use in the URI.
someLegal - characters that are legal in certain cases.
allLegal - characters that are always legal.

6.853.2.2 decaf::internal::net::URIHelper::URIHelper ()

Sets up the filter strings with sane defaults.

6.853.2.3 virtual decaf::internal::net::URIHelper::~~URIHelper () [inline, virtual]

6.853.3 Member Function Documentation

6.853.3.1 bool decaf::internal::net::URIHelper::isValidDomainName (const std::string & *host*)

Validates the string past to determine if it is a well formed domain name.

Parameters

host - domain name to validate.

Returns

true if host is well formed.

6.853.3.2 `bool decaf::internal::net::URIHelper::isValidHexChar (char c)`

Determines if the given char is a valid Hex char.

Valid chars are A-F (upper or lower case) and 0-9.

Parameters

c - char to inspect

Returns

true if c is a valid hex char.

6.853.3.3 `bool decaf::internal::net::URIHelper::isValidHost (bool forceServer, const std::string & host) throw (decaf::net::URISyntaxException)`

distinguish between IPv4, IPv6, domain name and validate it based on its type

Parameters

forceServer - true if the forceServer mode should be active.

host - Host string to validate.

Returns

true if the host value if a valid domain name.

Exceptions

URISyntaxException if the host is invalid and forceServer is true.

6.853.3.4 `bool decaf::internal::net::URIHelper::isValidIP4Word (const std::string & word)`

Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.

Parameters

word - string value to check.

Returns

true if the word is a valid IPv4 word.

6.853.3.5 bool decaf::internal::net::URIHelper::isValidIPv6Address (const std::string & *ipAddress*)

Determines if the given address is valid according to the IPv6 spec.

Parameters

ipAddress - string ip address value to validate.

Returns

true if the address string is valid.

6.853.3.6 bool decaf::internal::net::URIHelper::isValidIPv4Address (const std::string & *host*)

Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9.

and XXX is not greater than 255.

Parameters

host - IPv4 address string to parse.

Returns

true if host is a well formed IPv4 address.

6.853.3.7 URIType decaf::internal::net::URIHelper::parseAuthority (bool *forceServer*, const std::string & *authority*) throw (decaf::net::URISyntaxException)

determine the host, port and user-info if the authority parses successfully to a server based authority

behavior in error cases: if forceServer is true, throw URISyntaxException with the proper diagnostic messages. if forceServer is false assume this is a registry based uri, and just return leaving the host, port and user-info fields undefined.

and there are some error cases where URISyntaxException is thrown regardless of the forceServer parameter e.g. mal-formed ipv6 address

Parameters

forceServer

authority

Returns

a URIType (p. 3690) instance containing the parsed data.

Exceptions

URISyntaxException

6.853.3.8 `URIType decaf::internal::net::URIHelper::parseURI (const std::string & uri, bool forceServer) throw (decaf::net::URISyntaxException)`

Parse the passed in URI.

Parameters

uri - the URI to Parse

forceServer - if true invalid URI data throws an Exception

Returns

a `URIType` (p. 3690) instance containing the parsed data.

Exceptions

URISyntaxException if forceServer is true and the URI is invalid.

6.853.3.9 `void decaf::internal::net::URIHelper::validateAuthority (const std::string & uri, const std::string & authority, std::size_t index) throw (decaf::net::URISyntaxException)`

Validate that the URI Authority Segment contains no invalid encodings.

Parameters

uri - the full uri.

authority - the Authority to check.

index - position in the uri where Authority starts.

Exceptions

URISyntaxException if the fragment has errors.

6.853.3.10 `void decaf::internal::net::URIHelper::validateFragment (const std::string & uri, const std::string & fragment, std::size_t index) throw (decaf::net::URISyntaxException)`

Validate that the URI fragment contains no invalid encodings.

Parameters

uri - the full uri.

fragment - the fragment to check.

index - position in the uri where fragment starts.

Exceptions

URISyntaxException if the fragment has errors.

6.853.3.11 void decaf::internal::net::URIHelper::validatePath (const std::string & *uri*, const std::string & *path*, std::size_t *index*) throw (decaf::net::URISyntaxException)

Validate that the URI Path Segment contains no invalid encodings.

Parameters

uri - the full uri.

path - the path to check.

index - position in the uri where path starts.

Exceptions

URISyntaxException if the fragment has errors.

6.853.3.12 void decaf::internal::net::URIHelper::validateQuery (const std::string & *uri*, const std::string & *query*, std::size_t *index*) throw (decaf::net::URISyntaxException)

Validate that the URI Query Segment contains no invalid encodings.

Parameters

uri - the full uri.

query - the query to check.

index - position in the uri where fragment starts.

Exceptions

URISyntaxException if the fragment has errors.

6.853.3.13 void decaf::internal::net::URIHelper::validateScheme (const std::string & *uri*, const std::string & *scheme*, int *index*) throw (decaf::net::URISyntaxException)

Validate the schema portin of the URI.

Parameters

uri - the URI to check.

scheme - the schema section of the URI.

index - index in uri where schema starts.

Exceptions

URISyntaxException if the fragment has errors.

6.853.3.14 `void decaf::internal::net::URIHelper::validateSsp (const std::string & uri, const std::string & ssp, std::size_t index) throw (decaf::net::URISyntaxException)`

Validate that the URI Ssp Segment contains no invalid encodings.

Parameters

- uri* - the full uri.
- ssp* - the SSP to check.
- index* - position in the uri where Ssp starts.

Exceptions

URISyntaxException if the fragment has errors.

6.853.3.15 `void decaf::internal::net::URIHelper::validateUserinfo (const std::string & uri, const std::string & userinfo, std::size_t index) throw (decaf::net::URISyntaxException)`

Check the supplied user info for validity.

Parameters

- uri* - the uri to parse.
- userinfo* - supplied user info
- index* - index into the URI string where the data is located.

Returns

true if valid

Exceptions

URISyntaxException if an error occurs

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIHelper.h`

6.854 activemq::transport::failover::URIPool Class Reference

```
#include <src/main/activemq/transport/failover/URIPool.h>
```

Public Member Functions

- `URIPool ()`
Create an Empty URI Pool.

- **URIPool** (const decaf::util::List< URI > &uris)
Creates a new URI Pool using the given list as the initial Free List.
- **~URIPool** ()
- **URI getURI** () throw (decaf::lang::exceptions::NoSuchElementException)
Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a NoSuchElementException.
- void **addURI** (const URI &uri)
*Adds a URI to the free list, callers that have previously taken one using the **getURI** method should always return the URI when they close the resource that was connected to that URI.*
- void **addURIs** (const StlList< URI > &uris)
Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.
- void **removeURI** (const URI &uri)
Remove a given URI from the Free List.
- bool **isRandomize** () const
Is the URI that is given randomly picked from the pool or is each one taken in sequence.
- void **setRandomize** (bool value)
Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

6.854.1 Constructor & Destructor Documentation

6.854.1.1 activemq::transport::failover::URIPool::URIPool ()

Create an Empty URI Pool.

6.854.1.2 activemq::transport::failover::URIPool::URIPool (const decaf::util::List< URI > & *uris*)

Creates a new URI Pool using the given list as the initial Free List.

Parameters

uris - List of URI to place in the Pool.

6.854.1.3 activemq::transport::failover::URIPool::~~URIPool ()

6.854.2 Member Function Documentation

6.854.2.1 void activemq::transport::failover::URIPool::addURI (const URI & *uri*)

Adds a URI to the free list, callers that have previously taken one using the **getURI** method should always return the URI when they close the resource that was connected to that URI.

Parameters

uri - a URI previously taken from the pool.

6.854.2.2 void activemq::transport::failover::URIPool::addURIs (const StlList< URI > & *uris*)

Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.

Parameters

uris - List of URIs to add into the Pool.

6.854.2.3 URI activemq::transport::failover::URIPool::getURI () throw (decaf::lang::exceptions::NoSuchElementException)

Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a NoSuchElementException.

Receiving the exception is not an indication that a URI won't be available in the future, the caller should react accordingly.

Returns

the next free URI in the Pool.

Exceptions

NoSuchElementException if there are none free currently.

6.854.2.4 bool activemq::transport::failover::URIPool::isRandomize () const [inline]

Is the URI that is given randomly picked from the pool or is each one taken in sequence.

Returns

true if URI gets are random.

6.854.2.5 void activemq::transport::failover::URIPool::removeURI (const URI & *uri*)

Remove a given URI from the Free List.

Parameters

uri - the URI to find and remove from the free list

6.854.2.6 void activemq::transport::failover::URIPool::setRandomize (bool *value*) [inline]

Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

Parameters

value - true indicates URI gets are random.

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/URIPool.h

6.855 activemq::util::URISupport Class Reference

```
#include <src/main/activemq/util/URISupport.h>
```

Static Public Member Functions

- static void **parseURL** (const std::string &URI, decaf::util::Properties &properties) throw (decaf::lang::exceptions::IllegalArgumentException)

Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.

- static **CompositeData** **parseComposite** (const URI &uri) throw (decaf::net::URISyntaxException)

Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.

- static decaf::util::Properties **parseQuery** (std::string query) throw (decaf::lang::exceptions::IllegalArgumentException)

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

- static void **parseQuery** (std::string query, decaf::util::Properties *properties) throw (decaf::lang::exceptions::IllegalArgumentException)

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

- static std::string **createQueryString** (const Properties &options) throw (decaf::net::URISyntaxException)

Given a properties object create a string that can be appended to a URI as a valid Query string.

6.855.1 Member Function Documentation

6.855.1.1 `static std::string activemq::util::URISupport::createQueryString (const Properties & options) throw (decaf::net::URISyntaxException) [static]`

Given a properties object create a string that can be appended to a URI as a valid Query string.

Parameters

options Properties object containing key / value query values.

Returns

a valid URI query string.

Exceptions

URISyntaxException if the string in the Properties object can't be encoded into a valid URI Query string.

6.855.1.2 `static CompositeData activemq::util::URISupport::parseComposite (const URI & uri) throw (decaf::net::URISyntaxException) [static]`

Parses a Composite URI into a Composite Data instance, the Composite URI takes the form scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.

Parameters

uri - The Composite URI to parse.

Returns

a new **CompositeData** (p. 1129) object with the parsed data

Exceptions

URISyntaxException if the URI is not well formed.

6.855.1.3 `static void activemq::util::URISupport::parseQuery (std::string query, decaf::util::Properties * properties) throw (decaf::lang::exceptions::IllegalArgumentException) [static]`

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

Parameters

query - the query string to parse.

properties - object pointer to get the parsed output.

Exceptions

IllegalArgumentException if the Query string is not well formed.

6.855.1.4 static decaf::util::Properties activemq::util::URISupport::parseQuery (std::string *query*) throw (decaf::lang::exceptions::IllegalArgumentException) [static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

Parameters

query The query string to parse and extract the encoded properties.

Returns

Properties object with the parsed output.

Exceptions

IllegalArgumentException if the Query string is not well formed.

6.855.1.5 static void activemq::util::URISupport::parseURL (const std::string & *URI*, decaf::util::Properties & *properties*) throw (decaf::lang::exceptions::IllegalArgumentException) [static]

Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.

Parameters

URI a Broker URI to parse

properties a Properties object to set the parsed values in

Exceptions

IllegalArgumentException if the passed URI is invalid

The documentation for this class was generated from the following file:

- src/main/activemq/util/URISupport.h

6.856 decaf::net::URISyntaxException Class Reference

```
#include <src/main/decaf/net/URISyntaxException.h>
```

Inheritance diagram for decaf::net::URISyntaxException:

Public Member Functions

- **URISyntaxException** () throw ()
Default Constructor.

- **URISyntaxException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **URISyntaxException** (const **URISyntaxException** &ex) throw ()
Copy Constructor.
- **URISyntaxException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const std::exception *cause) throw ()
Constructor.
- **URISyntaxException** (const char *file, const int lineNumber, const char *msg DECAF_ - UNUSED) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const char *file, const int lineNumber, const std::string &input, const std::string &reason) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const char *file, const int lineNumber, const std::string &input, const std::string &reason, int index) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **URISyntaxException * clone** () const
Clones this exception.
- virtual **~URISyntaxException** () throw ()
- std::string **getInput** () const
- std::string **getReason** () const
- int **getIndex** () const

6.856.1 Constructor & Destructor Documentation

6.856.1.1 decaf::net::URISyntaxException::URISyntaxException () throw () [inline]

Default Constructor.

6.856.1.2 decaf::net::URISyntaxException::URISyntaxException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.856.1.3 decaf::net::URISyntaxException::URISyntaxException (const URISyntaxException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.856.1.4 decaf::net::URISyntaxException::URISyntaxException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.856.1.5 decaf::net::URISyntaxException::URISyntaxException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.856.1.6 decaf::net::URISyntaxException::URISyntaxException (const char * *file*, const int *lineNumber*, const char **msg* *DECAF_UNUSED*) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.856.1.7 `decaf::net::URISyntaxException::URISyntaxException (const char *
file, const int lineNumber, const std::string & input, const std::string
& reason) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the input string that caused the error and the reason for the error.

Parameters

file The file name where exception occurs.

lineNumber The line number where the exception occurred.

input The **URL** (p.3697) that caused the exception.

reason The reason for the failure.

6.856.1.8 `decaf::net::URISyntaxException::URISyntaxException (const char *
file, const int lineNumber, const std::string & input, const std::string
& reason, int index) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the input string that caused the error and the reason for the error.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

input The input **URI** (p.3660) that caused the exception

reason The reason for the failure.

index The index in the **URI** (p.3660) string where the error occurred.

6.856.1.9 `virtual decaf::net::URISyntaxException::~~URISyntaxException ()
throw () [inline, virtual]`

6.856.2 Member Function Documentation

6.856.2.1 `virtual URISyntaxException* decaf::net::URISyntaxException::clone (
) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::lang::Exception** (p.1715).

6.856.2.2 `int decaf::net::URISyntaxException::getIndex () const [inline]`**Returns**

the index in the input string where the error occurred or -1

6.856.2.3 `std::string decaf::net::URISyntaxException::getInput () const [inline]`**Returns**

the Input string that cause this exception or ""

6.856.2.4 `std::string decaf::net::URISyntaxException::getReason () const [inline]`**Returns**

the Reason given for this failure, or ""

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URISyntaxException.h`

6.857 decaf::internal::net::URIType Class Reference

Basic type object that holds data that composes a given URI.

```
#include <src/main/decaf/internal/net/URIType.h>
```

Public Member Functions

- **URIType** (const std::string &source)
- **URIType** ()
- virtual ~**URIType** ()
- std::string **getSource** () const
*Gets the source URI string that was parsed to obtain this **URIType** (p. 3690) instance and the resulting data,.*
- void **setSource** (const std::string &source)
*Sets the source URI string that was parsed to obtain this **URIType** (p. 3690) instance and the resulting data,.*
- std::string **getScheme** () const
Gets the Scheme of the URI, e.g.
- void **setScheme** (const std::string &scheme)
Sets the Scheme of the URI, e.g.
- std::string **getSchemeSpecificPart** () const

Gets the Scheme Specific Part of the URI.

- void **setSchemeSpecificPart** (const std::string &schemeSpecificPart)
Sets the Scheme Specific Part of the URI.
- std::string **getAuthority** () const
Gets the Authority of the URI.
- void **setAuthority** (const std::string &authority)
Sets the Authority of the URI.
- std::string **getUserInfo** () const
Gets the user info part of the URI, e.g.
- void **setUserInfo** (const std::string &userinfo)
Sets the user info part of the URI, e.g.
- std::string **getHost** () const
Gets the Host name part of the URI.
- void **setHost** (const std::string &host)
Sets the Host name part of the URI.
- int **getPort** () const
Gets the port part of the URI.
- void **setPort** (int port)
Sets the port part of the URI.
- std::string **getPath** () const
Gets the Path part of the URI.
- void **setPath** (const std::string &path)
Sets the Path part of the URI.
- std::string **getQuery** () const
Gets the Query part of the URI.
- void **setQuery** (const std::string &query)
Sets the Query part of the URI.
- std::string **getFragment** () const
Gets the Fragment part of the URI.
- void **setFragment** (const std::string &fragment)
Sets the Fragment part of the URI.
- bool **isOpaque** () const
Gets if the URI is Opaque.

- void **setOpaque** (bool opaque)
Sets if the URI is Opaque.
- bool **isAbsolute** () const
Gets if the URI is Absolute.
- void **setAbsolute** (bool absolute)
Sets if the URI is Absolute.
- bool **isServerAuthority** () const
Gets if the URI is a Server Authority.
- void **setServerAuthority** (bool serverAuthority)
Sets if the URI is a Server Authority.
- bool **isValid** () const
Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.
- void **setValid** (bool valid)
Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

6.857.1 Detailed Description

Basic type object that holds data that composes a given URI.

6.857.2 Constructor & Destructor Documentation

6.857.2.1 `decaf::internal::net::URIType::URIType (const std::string & source)`
[inline]

6.857.2.2 `decaf::internal::net::URIType::URIType ()` [inline]

6.857.2.3 `virtual decaf::internal::net::URIType::~~URIType ()` [inline, virtual]

6.857.3 Member Function Documentation

6.857.3.1 `std::string decaf::internal::net::URIType::getAuthority ()` const
[inline]

Gets the Authority of the URI.

Returns

Authority part string.

6.857.3.2 `std::string decaf::internal::net::URIType::getFragment () const` `[inline]`

Gets the Fragment part of the URI.

Returns

Fragment part string.

6.857.3.3 `std::string decaf::internal::net::URIType::getHost () const` `[inline]`

Gets the Host name part of the URI.

Returns

Host name part string.

6.857.3.4 `std::string decaf::internal::net::URIType::getPath () const` `[inline]`

Gets the Path part of the URI.

Returns

Path part string.

6.857.3.5 `int decaf::internal::net::URIType::getPort () const` `[inline]`

Gets the port part of the URI.

Returns

port part string, -1 if not set.

6.857.3.6 `std::string decaf::internal::net::URIType::getQuery () const` `[inline]`

Gets the Query part of the URI.

Returns

Query part string.

6.857.3.7 `std::string decaf::internal::net::URIType::getScheme () const` `[inline]`

Gets the Scheme of the URI, e.g.

scheme ("http"/"ftp"/...).

Returns

scheme part string.

6.857.3.8 `std::string decaf::internal::net::URIType::getSchemeSpecificPart () const [inline]`

Gets the Scheme Specific Part of the URI.

Returns

scheme specific part string.

6.857.3.9 `std::string decaf::internal::net::URIType::getSource () const [inline]`

Gets the source URI string that was parsed to obtain this **URIType** (p.3690) instance and the resulting data,.

Returns

the source URI string

6.857.3.10 `std::string decaf::internal::net::URIType::getUserInfo () const [inline]`

Gets the user info part of the URI, e.g.

user name, as in `http://user:passwd@host:port/`

Returns

user info part string.

6.857.3.11 `bool decaf::internal::net::URIType::isAbsolute () const [inline]`

Gets if the URI is Absolute.

Returns

true if Absolute.

6.857.3.12 `bool decaf::internal::net::URIType::isOpaque () const [inline]`

Gets if the URI is Opaque.

Returns

true if opaque.

6.857.3.13 `bool decaf::internal::net::URIType::isServerAuthority () const [inline]`

Gets if the URI is a Server Authority.

Returns

true if Server Authority.

6.857.3.14 `bool decaf::internal::net::URIType::isValid () const [inline]`

Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

Returns

true if the **URIType** (p. 3690) contains valid data.

6.857.3.15 `void decaf::internal::net::URIType::setAbsolute (bool absolute) [inline]`

Sets if the URI is Absolute.

Parameters

absolute - true if Absolute.

6.857.3.16 `void decaf::internal::net::URIType::setAuthority (const std::string & authority) [inline]`

Sets the Authority of the URI.

Parameters

authority Authority part string.

6.857.3.17 `void decaf::internal::net::URIType::setFragment (const std::string & fragment) [inline]`

Sets the Fragment part of the URI.

Parameters

fragment - Fragment part string.

6.857.3.18 `void decaf::internal::net::URIType::setHost (const std::string & host) [inline]`

Sets the Host name part of the URI.

Parameters

host - Host name part string.

6.857.3.19 `void decaf::internal::net::URIType::setOpaque (bool opaque) [inline]`

Sets if the URI is Opaque.

Parameters

opaque true if opaque.

6.857.3.20 void decaf::internal::net::URIType::setPath (const std::string & *path*) [inline]

Sets the Path part of the URL.

Parameters

path - Path part string.

6.857.3.21 void decaf::internal::net::URIType::setPort (int *port*) [inline]

Sets the port part of the URL.

Parameters

port - port part string, -1 if not set.

6.857.3.22 void decaf::internal::net::URIType::setQuery (const std::string & *query*) [inline]

Sets the Query part of the URL.

Parameters

query - Query part string.

6.857.3.23 void decaf::internal::net::URIType::setScheme (const std::string & *scheme*) [inline]

Sets the Scheme of the URL, e.g.

scheme ("http"/"ftp"/...).

Parameters

scheme - scheme part string.

6.857.3.24 void decaf::internal::net::URIType::setSchemeSpecificPart (const std::string & *schemeSpecificPart*) [inline]

Sets the Scheme Specific Part of the URL.

Parameters

schemeSpecificPart - scheme specific part string.

6.857.3.25 `void decaf::internal::net::URIType::setServerAuthority (bool
 serverAuthority) [inline]`

Sets if the URI is a Server Authority.

Parameters

serverAuthority - true if Server Authority.

6.857.3.26 `void decaf::internal::net::URIType::setSource (const std::string &
 source) [inline]`

Sets the source URI string that was parsed to obtain this **URIType** (p. 3690) instance and the resulting data,.

Parameters

source - the source URI string

6.857.3.27 `void decaf::internal::net::URIType::setUserInfo (const std::string &
 userinfo) [inline]`

Sets the user info part of the URI, e.g.

user name, as in `http://user:passwd@host:port/`

Parameters

userinfo - user info part string.

6.857.3.28 `void decaf::internal::net::URIType::setValid (bool valid) [inline]`

Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

Parameters

valid - true if the **URIType** (p. 3690) contains valid data.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIType.h`

6.858 decaf::net::URL Class Reference

Class **URL** (p. 3697) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.

`#include <src/main/decaf/net/URL.h>`

Public Member Functions

- **URL** ()
- **URL** (const std::string &url)
- virtual ~**URL** ()

6.858.1 Detailed Description

Class **URL** (p. 3697) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web. A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object, such as a query to a database or to a search engine. More information on the types of URLs and their formats can be found at:

`http://www.ksc.nasa.gov/facts/internet/url-primer.html`

In general, a **URL** (p. 3697) can be broken into several parts. The previous example of a **URL** (p. 3697) indicates that the protocol to use is http (HyperText Transfer Protocol) and that the information resides on a host machine named `www.ksc.nasa.gov`. The information on that host machine is named `/facts/internet/url-primer.html`. The exact meaning of this name on the host machine is both protocol dependent and host dependent. The information normally resides in a file, but it could be generated on the fly. This component of the **URL** (p. 3697) is called the path component.

A **URL** (p. 3697) can optionally specify a "port", which is the port number to which the TCP connection is made on the remote host machine. If the port is not specified, the default port for the protocol is used instead. For example, the default port for http is 80. An alternative port could be specified as:

`http://www.ksc.nasa.gov:80/facts/internet/url-primer.html`

The syntax of **URL** (p. 3697) is defined by RFC 2396: Uniform Resource Identifiers (**URI** (p. 3660)): Generic Syntax, amended by RFC 2732: Format for Literal IPv6 Addresses in URLs. The Literal IPv6 address format also supports `scope_ids`. The syntax and usage of `scope_ids` is described here.

A **URL** (p. 3697) may have appended to it a "fragment", also known as a "ref" or a "reference". The fragment is indicated by the sharp sign character "#" followed by more characters. For example,

`http://www.apache.org/cms/index.html#chapter1`

This fragment is not technically part of the **URL** (p. 3697). Rather, it indicates that after the specified resource is retrieved, the application is specifically interested in that part of the document that has the tag `chapter1` attached to it. The meaning of a tag is resource specific.

An application can also specify a "relative URL", which contains only enough information to reach the resource relative to another **URL** (p. 3697). Relative URLs are frequently used within HTML pages. For example, if the contents of the **URL** (p. 3697):

`http://www.apache.org/cms/index.html`

contained within it the relative **URL** (p. 3697):

`FAQ.html`

it would be a shorthand for:

`http://www.apache.org/cms/FAQ.html`

The relative **URL** (p. 3697) need not specify all the components of a **URL** (p. 3697). If the protocol, host name, or port number is missing, the value is inherited from the fully specified

URL (p. 3697). The file component must be specified. The optional fragment is not inherited.

The **URL** (p. 3697) class does not itself encode or decode any **URL** (p. 3697) components according to the escaping mechanism defined in RFC2396. It is the responsibility of the caller to encode any fields, which need to be escaped prior to calling **URL** (p. 3697), and also to decode any escaped fields, that are returned from **URL** (p. 3697). Furthermore, because **URL** (p. 3697) has no knowledge of **URL** (p. 3697) escaping, it does not recognise equivalence between the encoded or decoded form of the same **URL** (p. 3697). For example, the two URLs:

`http://foo.com/hello world/` and `http://foo.com/hello%20world`

would be considered not equal to each other.

Note, the **URI** (p. 3660) class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use **URI** (p. 3660), and to convert between these two classes using `toURI()` and `URI.toURL()` (p. 3671).

The **URLEncoder** (p. 3700) and **URLDecoder** (p. 3699) classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

6.858.2 Constructor & Destructor Documentation

6.858.2.1 `decaf::net::URL::URL ()`

6.858.2.2 `decaf::net::URL::URL (const std::string & url)`

6.858.2.3 `virtual decaf::net::URL::~~URL ()` [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URL.h`

6.859 decaf::net::URLDecoder Class Reference

```
#include <src/main/decaf/net/URLDecoder.h>
```

Public Member Functions

- `virtual ~URLDecoder ()`

Static Public Member Functions

- `static std::string decode (const std::string &value)`

Decodes the string argument which is assumed to be encoded in the `x-www-form-urlencoded` MIME content type.

6.859.1 Constructor & Destructor Documentation

6.859.1.1 `virtual decaf::net::URLDecoder::~~URLDecoder ()` [inline, virtual]

6.859.2 Member Function Documentation

6.859.2.1 `static std::string decaf::net::URLDecoder::decode (const std::string & value)` [static]

Decodes the string argument which is assumed to be encoded in the `x-www-form-urlencoded` MIME content type.

'+' will be converted to space, '%' and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A+B+C %24%25" -> "A B C \$%"

Parameters

value - string The encoded string.

Returns

The decoded version as a string.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLDecoder.h`

6.860 decaf::net::URLEncoder Class Reference

```
#include <src/main/decaf/net/URLEncoder.h>
```

Public Member Functions

- `virtual ~URLEncoder ()`

Static Public Member Functions

- `static std::string encode (const std::string &value)`

This class contains a utility method for converting a string to the format required by the application/x-www-form-urlencoded MIME content type.

6.860.1 Constructor & Destructor Documentation

6.860.1.1 `virtual decaf::net::URLEncoder::~~URLEncoder () [inline, virtual]`

6.860.2 Member Function Documentation

6.860.2.1 `static std::string decaf::net::URLEncoder::encode (const std::string & value) [static]`

This class contains a utility method for converting a string to the format required by the application/x-www-form-urlencoded MIME content type.

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and characters '.', '-', '*', '_' are converted into their hexadecimal value prepended by ' '.

For example: '#' -> '23'

In addition, spaces are substituted by '+'

Parameters

value - the string to be converted

Returns

the converted string

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URLEncoder.h**

6.861 activemq::util::Usage Class Reference

```
#include <src/main/activemq/util/Usage.h>
```

Inheritance diagram for activemq::util::Usage:

Public Member Functions

- virtual `~Usage ()`
- virtual void `waitForSpace ()=0`
*Waits forever for more space to be returned to this **Usage** (p. 3701) Manager.*
- virtual void `waitForSpace (unsigned int timeout)=0`
*Waits for more space to be returned to this **Usage** (p. 3701) Manager, times out when the given time span in milliseconds elapses.*
- virtual void `enqueueUsage (unsigned long long value)=0`
Tries to increase the usage by value amount but blocks if this object is currently full.
- virtual void `increaseUsage (unsigned long long value)=0`

Increases the usage by the value amount.

- virtual void **decreaseUsage** (unsigned long long value)=0

Decreases the usage by the value amount.

- virtual bool **isFull** () const =0

*Returns true if this **Usage** (p. 3701) instance is full, i.e.*

6.861.1 Constructor & Destructor Documentation

6.861.1.1 virtual **activemq::util::Usage::~Usage** () [inline, virtual]

6.861.2 Member Function Documentation

6.861.2.1 virtual void **activemq::util::Usage::decreaseUsage** (unsigned long long *value*) [pure virtual]

Decreases the usage by the value amount.

Parameters

value Amount of space to return to the pool

Implemented in **activemq::util::MemoryUsage** (p. 2356).

6.861.2.2 virtual void **activemq::util::Usage::enqueueUsage** (unsigned long long *value*) [pure virtual]

Tries to increase the usage by value amount but blocks if this object is currently full.

Parameters

value Amount of usage in bytes to add.

Implemented in **activemq::util::MemoryUsage** (p. 2356).

6.861.2.3 virtual void **activemq::util::Usage::increaseUsage** (unsigned long long *value*) [pure virtual]

Increases the usage by the value amount.

Parameters

value Amount of usage to add.

Implemented in **activemq::util::MemoryUsage** (p. 2357).

6.861.2.4 virtual bool activemq::util::Usage::isFull () const [pure virtual]

Returns true if this **Usage** (p. 3701) instance is full, i.e.

Usage (p. 3701) $\geq 100\%$

Returns

true if **Usage** (p. 3701) is at the Full point.

Implemented in **activemq::util::MemoryUsage** (p. 2357).

6.861.2.5 virtual void activemq::util::Usage::waitForSpace (unsigned int *timeout*) [pure virtual]

Waits for more space to be returned to this **Usage** (p. 3701) Manager, times out when the given time span in milliseconds elapses.

Parameters

timeout The time to wait for more space.

Implemented in **activemq::util::MemoryUsage** (p. 2357).

6.861.2.6 virtual void activemq::util::Usage::waitForSpace () [pure virtual]

Waits forever for more space to be returned to this **Usage** (p. 3701) Manager.

Implemented in **activemq::util::MemoryUsage** (p. 2357).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**Usage.h**

6.862 decaf::io::UTFDataFormatException Class Reference

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

```
#include <src/main/decaf/io/UTFDataFormatException.h>
```

Inheritance diagram for decaf::io::UTFDataFormatException:

Public Member Functions

- **UTFDataFormatException** () throw ()
Default Constructor.
- **UTFDataFormatException** (const lang::Exception &ex) throw ()
Copy Constructor.

- **UTFDataFormatException** (const **UTFDataFormatException** &ex) throw ()
Copy Constructor.
- **UTFDataFormatException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UTFDataFormatException** (const std::exception *cause) throw ()
Constructor.
- **UTFDataFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **UTFDataFormatException** * clone () const
Clones this exception.
- virtual ~**UTFDataFormatException** () throw ()

6.862.1 Detailed Description

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Since

1.0

6.862.2 Constructor & Destructor Documentation

6.862.2.1 decaf::io::UTFDataFormatException::UTFDataFormatException () throw () [inline]

Default Constructor.

6.862.2.2 decaf::io::UTFDataFormatException::UTFDataFormatException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.862.2.3 decaf::io::UTFDataFormatException::UTFDataFormatException (const UTFDataFormatException & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.862.2.4 `decaf::io::UTFDataFormatException::UTFDataFormatException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- cause* The exception that was the cause for this one to be thrown.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.862.2.5 `decaf::io::UTFDataFormatException::UTFDataFormatException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

- cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.862.2.6 `decaf::io::UTFDataFormatException::UTFDataFormatException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.862.2.7 `virtual decaf::io::UTFDataFormatException::~~UTFDataFormatException () throw () [inline, virtual]`

6.862.3 Member Function Documentation

6.862.3.1 `virtual UTFDataFormatException* decaf::io::UTFDataFormatException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 2005).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/UTFDataFormatException.h`

6.863 decaf::util::UUID Class Reference

A class that represents an immutable universally unique identifier (**UUID** (p. 3706)).

```
#include <src/main/decaf/util/UUID.h>
```

Inheritance diagram for `decaf::util::UUID`:

Public Member Functions

- **UUID** (long long mostSigBits, long long leastSigBits)
*Constructs a new **UUID** (p. 3706) using the specified data.*
- virtual **~UUID** ()
- virtual int **compareTo** (const **UUID** &value) const
*Compare the given **UUID** (p. 3706) to this one.*
- virtual bool **equals** (const **UUID** &value) const
*Compares this **UUID** (p. 3706) to the one given, returns true if they are equal.*
- virtual bool **operator==** (const **UUID** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **UUID** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual std::string **toString** () const
*Returns a String object representing this **UUID** (p. 3706).*
- virtual long long **getLeastSignificantBits** () const
- virtual long long **getMostSignificantBits** () const
- virtual long long **node** () throw (lang::exceptions::UnsupportedOperationException)
*The node value associated with this **UUID** (p. 3706).*
- virtual long long **timestamp** () throw (lang::exceptions::UnsupportedOperationException)
*The timestamp value associated with this **UUID** (p. 3706).*

- virtual int **clockSequence** () throw (lang::exceptions::UnsupportedOperationException)
*The clock sequence value associated with this **UUID** (p. 3706).*
- virtual int **variant** () throw (lang::exceptions::UnsupportedOperationException)
*The variant number associated with this **UUID** (p. 3706).*
- virtual int **version** () throw (lang::exceptions::UnsupportedOperationException)
*The version number associated with this **UUID** (p. 3706).*

Static Public Member Functions

- static **UUID randomUUID** ()
*Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 3706).*
- static **UUID nameUUIDFromBytes** (const std::vector< char > &name)
*Static factory to retrieve a type 3 (name based) **UUID** (p. 3706) based on the specified byte array.*
- static **UUID nameUUIDFromBytes** (const char *name, std::size_t size)
*Static factory to retrieve a type 3 (name based) **UUID** (p. 3706) based on the specified byte array.*
- static **UUID fromString** (const std::string &name) throw (lang::exceptions::IllegalArgumentException)
*Creates a **UUID** (p. 3706) from the string standard representation as described in the **toString()** (p. 3711) method.*

6.863.1 Detailed Description

A class that represents an immutable universally unique identifier (**UUID** (p. 3706)). A **UUID** (p. 3706) represents a 128-bit value.

There exist different variants of these global identifiers. The methods of this class are for manipulating the Leach-Salz variant, although the constructors allow the creation of any variant of **UUID** (p. 3706) (described below).

The layout of a variant 2 (Leach-Salz) **UUID** (p. 3706) is as follows: The most significant long consists of the following unsigned fields:

```
0xFFFFFFFF00000000 time_low 0x00000000FFFF0000 time_mid 0x000000000000F000 version
0x00000000000000FF time_hi
```

The least significant long consists of the following unsigned fields:

```
0xC000000000000000 variant 0x3FFF000000000000 clock_seq 0x0000FFFFFFFFFFFFFF node
```

The variant field contains a value which identifies the layout of the **UUID** (p. 3706). The bit layout described above is valid only for a **UUID** (p. 3706) with a variant value of 2, which indicates the Leach-Salz variant.

The version field holds a value that describes the type of this **UUID** (p. 3706). There are four different basic types of UUIDs: time-based, DCE security, name-based, and randomly generated UUIDs. These types have a version value of 1, 2, 3 and 4, respectively.

For more information including algorithms used to create UUIDs, see the Internet-Draft UUIDs and GUIDs or the standards body definition at ISO/IEC 11578:1996.

6.863.2 Constructor & Destructor Documentation

6.863.2.1 decaf::util::UUID::UUID (long long *mostSigBits*, long long *leastSigBits*)

Constructs a new **UUID** (p. 3706) using the specified data.

mostSigBits is used for the most significant 64 bits of the **UUID** (p. 3706) and *leastSigBits* becomes the least significant 64 bits of the **UUID** (p. 3706).

Parameters

mostSigBits

leastSigBits

6.863.2.2 virtual decaf::util::UUID::~~UUID () [virtual]

6.863.3 Member Function Documentation

6.863.3.1 virtual int decaf::util::UUID::clockSequence () throw (lang::exceptions::UnsupportedOperationException) [virtual]

The clock sequence value associated with this **UUID** (p. 3706).

The 14 bit clock sequence value is constructed from the clock sequence field of this **UUID** (p. 3706). The clock sequence field is used to guarantee temporal uniqueness in a time-based **UUID** (p. 3706).

The *clockSequence* value is only meaningful in a time-based **UUID** (p. 3706), which has version type 1. If this **UUID** (p. 3706) is not a time-based **UUID** (p. 3706) then this method throws *UnsupportedOperationException*.

Returns

the *clockSequence* associated with a V1 **UUID** (p. 3706)

Exceptions

UnsupportedOperationException

6.863.3.2 virtual int decaf::util::UUID::compareTo (const UUID & *value*) const [virtual]

Compare the given **UUID** (p. 3706) to this one.

Parameters

value - the **UUID** (p. 3706) to compare to

6.863.3.3 `virtual bool decaf::util::UUID::equals (const UUID & value) const`
[virtual]

Compares this **UUID** (p. 3706) to the one given, returns true if they are equal.

Parameters

value - the **UUID** (p. 3706) to compare to.

Returns

true if UUIDs are the same.

6.863.3.4 `static UUID decaf::util::UUID::fromString (const std::string & name)`
`throw (lang::exceptions::IllegalArgumentException)` [static]

Creates a **UUID** (p. 3706) from the string standard representation as described in the `toString()` (p. 3711) method.

Parameters

name - a string to be used to construct a **UUID** (p. 3706).

Returns

type 3 **UUID** (p. 3706)

6.863.3.5 `virtual long long decaf::util::UUID::getLeastSignificantBits () const`
[virtual]

Returns

the most significant 64 bits of this UUID's 128 bit value.

6.863.3.6 `virtual long long decaf::util::UUID::getMostSignificantBits () const`
[virtual]

Returns

the most significant 64 bits of this UUID's 128 bit value.

6.863.3.7 `static UUID decaf::util::UUID::nameUUIDFromBytes (const`
`std::vector< char > & name)` [static]

Static factory to retrieve a type 3 (name based) **UUID** (p. 3706) based on the specified byte array.

Parameters

name - a byte array to be used to construct a **UUID** (p. 3706).

Returns

type 3 **UUID** (p. 3706)

```
6.863.3.8 static UUID decaf::util::UUID::nameUUIDFromBytes ( const char *
name, std::size_t size ) [static]
```

Static factory to retrieve a type 3 (name based) **UUID** (p. 3706) based on the specified byte array.

Parameters

name - a byte array to be used to construct a **UUID** (p. 3706).

size - the size of the byte array, or number of bytes to use.

Returns

type 3 **UUID** (p. 3706)

```
6.863.3.9 virtual long long decaf::util::UUID::node ( ) throw (
lang::exceptions::UnsupportedOperationException ) [virtual]
```

The node value associated with this **UUID** (p. 3706).

The 48 bit node value is constructed from the node field of this **UUID** (p.3706). This field is intended to hold the IEEE 802 address of the machine that generated this **UUID** (p.3706) to guarantee spatial uniqueness.

The node value is only meaningful in a time-based **UUID** (p. 3706), which has version type 1. If this **UUID** (p. 3706) is not a time-based **UUID** (p. 3706) then this method throws `UnsupportedOperationException`.

Returns

the node value of this **UUID** (p. 3706)

Exceptions

UnsupportedOperationException

```

6.863.3.10 virtual bool decaf::util::UUID::operator< ( const UUID & value )
const [virtual]

```

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.863.3.11 `virtual bool decaf::util::UUID::operator==(const UUID & value) const [virtual]`

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.863.3.12 `static UUID decaf::util::UUID::randomUUID () [static]`

Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 3706).

The **UUID** (p. 3706) is generated using a cryptographically strong pseudo random number generator.

Returns

type 4 **UUID** (p. 3706)

6.863.3.13 `virtual long long decaf::util::UUID::timestamp () throw (lang::exceptions::UnsupportedOperationException) [virtual]`

The timestamp value associated with this **UUID** (p. 3706).

The 60 bit timestamp value is constructed from the time_low, time_mid, and time_hi fields of this **UUID** (p. 3706). The resulting timestamp is measured in 100-nanosecond units since midnight, October 15, 1582 UTC.

The timestamp value is only meaningful in a time-based **UUID** (p. 3706), which has version type 1. If this **UUID** (p. 3706) is not a time-based **UUID** (p. 3706) then this method throws `UnsupportedOperationException`.

Returns

the timestamp associated with a V1 **UUID** (p. 3706)

Exceptions

UnsupportedOperationException

6.863.3.14 `virtual std::string decaf::util::UUID::toString () const [virtual]`

Returns a String object representing this **UUID** (p. 3706).

UUID's are formatted as: 00112233-4455-6677-8899-AABBCCDDEEFF whose length is 36.

Returns

formatted string for this **UUID** (p. 3706)

6.863.3.15 `virtual int decaf::util::UUID::variant () throw (lang::exceptions::UnsupportedOperationException) [virtual]`

The variant number associated with this **UUID** (p. 3706).

The variant number describes the layout of the **UUID** (p. 3706). The variant number has the following meaning:

* 0 Reserved for NCS backward compatibility * 2 The Leach-Salz variant (used by this class) * 6 Reserved, Microsoft Corporation backward compatibility * 7 Reserved for future definition

Returns

the variant associated with a V1 **UUID** (p. 3706)

Exceptions

UnsupportedOperationException

6.863.3.16 `virtual int decaf::util::UUID::version () throw (lang::exceptions::UnsupportedOperationException) [virtual]`

The version number associated with this **UUID** (p. 3706).

The version number describes how this **UUID** (p. 3706) was generated. The version number has the following meaning:

* 1 Time-based **UUID** (p. 3706) * 2 DCE security **UUID** (p. 3706) * 3 Name-based **UUID** (p. 3706) * 4 Randomly generated **UUID** (p. 3706)

Returns

the version associated with a V1 **UUID** (p. 3706)

Exceptions

UnsupportedOperationException

The documentation for this class was generated from the following file:

- `src/main/decaf/util/UUID.h`

6.864 activemq::wireformat::WireFormat Class Reference

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

```
#include <src/main/activemq/wireformat/WireFormat.h>
```

Inheritance diagram for `activemq::wireformat::WireFormat`:

Public Member Functions

- virtual `~WireFormat ()`
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out)=0
throw (**decaf::io::IOException**)

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in)=0
throw (**decaf::io::IOException**)

Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

- virtual void **setVersion** (int version)=0

Set the Version.

- virtual int **getVersion** () const =0

Get the Version.

- virtual bool **hasNegotiator** () const =0

*Returns true if this **WireFormat** (p. 3712) has a Negotiator that needs to wrap the Transport that uses it.*

- virtual bool **inReceive** () const =0

Indicates if the WireFormat object is in the process of receiving a message.

- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport)=0 throw (**decaf::lang::exceptions::UnsupportedOperationException**)

If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

6.864.1 Detailed Description

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

Version

Revision:

1.1

6.864.2 Constructor & Destructor Documentation

6.864.2.1 virtual `activemq::wireformat::WireFormat::~~WireFormat ()` [inline, virtual]

6.864.3 Member Function Documentation

6.864.3.1 virtual `Pointer<transport::Transport> activemq::wireformat::WireFormat::createNegotiator (const Pointer< transport::Transport > & transport) throw (decaf::lang::exceptions::UnsupportedOperationException)` [pure virtual]

If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

Parameters

transport - the Transport to Wrap the Negotiator around.

Returns

new instance of a **WireFormatNegotiator** (p. 3751) as a **Pointer<Transport>** (p. 2756).

Exceptions

UnsupportedOperationException if the **WireFormat** (p. 3712) doesn't have a Negotiator.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2703), and **activemq::wireformat::stomp::StompWireFormat** (p. 3408).

6.864.3.2 virtual `int activemq::wireformat::WireFormat::getVersion () const` [pure virtual]

Get the Version.

Returns

the version of the wire format

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2705), and **activemq::wireformat::stomp::StompWireFormat** (p. 3408).

6.864.3.3 virtual `bool activemq::wireformat::WireFormat::hasNegotiator () const` [pure virtual]

Returns true if this **WireFormat** (p. 3712) has a Negotiator that needs to wrap the Transport that uses it.

Returns

true if the **WireFormat** (p. 3712) provides a Negotiator.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2705), and **activemq::wireformat::stomp::StompWireFormat** (p. 3409).

6.864.3.4 `virtual bool activemq::wireformat::WireFormat::inReceive () const` [pure virtual]

Indicates if the WireFormat object is in the process of receiving a message.

This is useful for monitoring inactivity and the **WireFormat** (p. 3712) is processing a large message which takes longer than some configured timeout to unmarshal, the inactivity monitor can query the **WireFormat** (p. 3712) instance to determine if its busy or not and not mark the connection as inactive if so.

Returns

true if the **WireFormat** (p. 3712) object is unmarshaling a message.

Implemented in `activemq::wireformat::openwire::OpenWireFormat` (p. 2705), and `activemq::wireformat::stomp::StompWireFormat` (p. 3409).

6.864.3.5 `virtual void activemq::wireformat::WireFormat::marshal` (`const Pointer< commands::Command > & command,` `const activemq::transport::Transport * transport,` `decaf::io::DataOutputStream * out`) `throw (decaf::io::IOException)` [pure virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters

command The Command to Marshal

transport The Transport that called this method.

out The output stream to write the command to.

Exceptions

IOException

Implemented in `activemq::wireformat::openwire::OpenWireFormat` (p. 2707), and `activemq::wireformat::stomp::StompWireFormat` (p. 3409).

6.864.3.6 `virtual void activemq::wireformat::WireFormat::setVersion (int version`) [pure virtual]

Set the Version.

Parameters

version the version of the wire format

Implemented in `activemq::wireformat::openwire::OpenWireFormat` (p. 2710), and `activemq::wireformat::stomp::StompWireFormat` (p. 3409).

6.864.3.7 virtual `Pointer<commands::Command>` `activemq::wireformat::WireFormat::unmarshal` (const `activemq::transport::Transport` * *transport*, `decaf::io::DataInputStream` * *in*) throw (`decaf::io::IOException`) [pure virtual]

Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

Returns a Pointer to the newly unmarshaled Command.

Parameters

- transport* - Pointer to the transport that is making this request.
- in* - the input stream to read the command from.

Returns

the newly marshaled Command, caller owns the pointer

Exceptions

IOException

Implemented in `activemq::wireformat::openwire::OpenWireFormat` (p. 2711), and `activemq::wireformat::stomp::StompWireFormat` (p. 3410).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/WireFormat.h`

6.865 activemq::wireformat::WireFormatFactory Class Reference

The `WireFormatFactory` (p. 3716) is the interface that all `WireFormatFactory` (p. 3716) classes must extend.

```
#include <src/main/activemq/wireformat/WireFormatFactory.h>
```

Inheritance diagram for `activemq::wireformat::WireFormatFactory`:

Public Member Functions

- virtual `~WireFormatFactory` ()
- virtual `Pointer< WireFormat >` `createWireFormat` (const `decaf::util::Properties` &properties)=0 throw (`decaf::lang::exceptions::IllegalStateException`)

Creates a new `WireFormat` (p. 3712) Object passing it a set of properties from which it can obtain any optional settings.

6.865.1 Detailed Description

The **WireFormatFactory** (p. 3716) is the interface that all **WireFormatFactory** (p. 3716) classes must extend. The Factory creates a **WireFormat** (p. 3712) Object based on the properties that are set in the passed **Properties** object.

6.865.2 Constructor & Destructor Documentation

6.865.2.1 `virtual activemq::wireformat::WireFormatFactory::~~WireFormatFactory () [inline, virtual]`

6.865.3 Member Function Documentation

6.865.3.1 `virtual Pointer<WireFormat> activemq::wireformat::WireFormatFactory::createWireFormat (const decaf::util::Properties & properties) throw (decaf::lang::exceptions::IllegalStateException) [pure virtual]`

Creates a new **WireFormat** (p. 3712) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

properties - the Properties for this **WireFormat** (p. 3712)

Returns

Pointer to a new instance of a **WireFormat** (p. 3712) object.

Implemented in **activemq::wireformat::openwire::OpenWireFormatFactory** (p. 2713), and **activemq::wireformat::stomp::StompWireFormatFactory** (p. 3411).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/WireFormatFactory.h`

6.866 activemq::commands::WireFormatInfo Class Reference

```
#include <src/main/activemq/commands/WireFormatInfo.h>
```

Inheritance diagram for `activemq::commands::WireFormatInfo`:

Public Member Functions

- **WireFormatInfo** ()
- virtual **~WireFormatInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.

- virtual **DataSet** * **cloneDataSet** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataSet** (const **DataSet** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataSet** (p. 1553) such as its type and value of its elements.*
- virtual bool **equals** (const **DataSet** *value) const
*Compares the **DataSet** (p. 1553) passed in to this one, and returns if they are equivalent.*
- virtual bool **isMarshalAware** () const
Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- int **getVersion** () const
Get the current Wireformat Version.
- void **setVersion** (int version)
Set the current Wireformat Version.
- long long **getMaxInactivityDuration** () const
Returns the currently configured Max Inactivity duration.
- void **setMaxInactivityDuration** (long long maxInactivityDuration)
Sets the Max inactivity duration value.
- long long **getMaxInactivityDurationInitialDelay** () const
Returns the currently configured Max Inactivity Initial Delay duration.
- void **setMaxInactivityDurationInitialDelay** (long long maxInactivityDurationInitialDelay)
Sets the Max inactivity initial delay duration value.
- bool **isStackTraceEnabled** () const
Checks if the stackTraceEnabled flag is on.
- void **setStackTraceEnabled** (bool stackTraceEnabled)
Sets if the stackTraceEnabled flag is on.
- bool **isTcpNoDelayEnabled** () const

Checks if the tcpNoDelayEnabled flag is on.

- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)
Sets if the tcpNoDelayEnabled flag is on.
- bool **isCacheEnabled** () const
Checks if the cacheEnabled flag is on.
- void **setCacheEnabled** (bool cacheEnabled)
Sets if the cacheEnabled flag is on.
- int **getCacheSize** () const
Gets the Cache Size setting.
- void **setCacheSize** (int value)
Sets the Cache Size setting.
- bool **isTightEncodingEnabled** () const
Checks if the tightEncodingEnabled flag is on.
- void **setTightEncodingEnabled** (bool tightEncodingEnabled)
Sets if the tightEncodingEnabled flag is on.
- bool **isSizePrefixDisabled** () const
Checks if the sizePrefixDisabled flag is on.
- void **setSizePrefixDisabled** (bool sizePrefixDisabled)
Sets if the sizePrefixDisabled flag is on.
- const std::vector< unsigned char > & **getMagic** () const
Get the Magic field.
- void **setMagic** (const std::vector< unsigned char > &magic)
Sets the value of the magic field.
- const std::vector< unsigned char > & **getMarshaledProperties** () const
Get the marshalledProperties field.
- void **setMarshaledProperties** (const std::vector< unsigned char > &marshalledProperties)
Sets the value of the marshalledProperties field.
- virtual const **util::PrimitiveMap** & **getProperties** () const
*Gets the Properties for this **Command** (p. 1107).*
- virtual **util::PrimitiveMap** & **getProperties** ()
*Gets the Properties for this **Command** (p. 1107).*
- virtual void **setProperties** (const **util::PrimitiveMap** &map)
*Sets the Properties for this **Command** (p. 1107).*

- bool **isValid** () const
*Determines if we think this is a Valid **WireFormatInfo** (p. 3717) command.*
- virtual bool **isWireFormatInfo** () const
- virtual void **beforeMarshal** (wireformat::WireFormat *wireFormat AMQCPP_ - UNUSED) throw (decaf::io::IOException)
Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.
- virtual void **afterUnmarshal** (wireformat::WireFormat *wireFormat AMQCPP_ - UNUSED) throw (decaf::io::IOException)
Called after unmarshaling is started to cleanup the object being unmarshaled.

Static Public Attributes

- static const unsigned char **ID_WIREFORMATINFO** = 1

6.866.1 Constructor & Destructor Documentation

6.866.1.1 activemq::commands::WireFormatInfo::WireFormatInfo ()

6.866.1.2 virtual activemq::commands::WireFormatInfo::~~WireFormatInfo ()
[virtual]

6.866.2 Member Function Documentation

6.866.2.1 virtual void activemq::commands::WireFormatInfo::afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_ UNUSED) throw (decaf::io::IOException) [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters

wireFormat - the wireformat object to control unmarshaling

Reimplemented from **activemq::commands::BaseDataStructure** (p. 765).

6.866.2.2 virtual void activemq::commands::WireFormatInfo::beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_ UNUSED) throw (decaf::io::IOException) [virtual]

Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.

Parameters

wireFormat - the wire formatting controller

Reimplemented from **activemq::commands::BaseDataStructure** (p. 765).

6.866.2.3 `virtual DataStructure* activemq::commands::WireFormatInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1554).

6.866.2.4 `virtual void activemq::commands::WireFormatInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.866.2.5 `virtual bool activemq::commands::WireFormatInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1553) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 696).

6.866.2.6 `int activemq::commands::WireFormatInfo::getCacheSize () const`

Gets the Cache Size setting.

Returns

currently set cache size.

6.866.2.7 `virtual unsigned char activemq::commands::WireFormatInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1553) type copy.

Implements **activemq::commands::DataSet** (p. 1557).

6.866.2.8 `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMagic () const [inline]`

Get the Magic field.

Returns

const reference to a `std::vector<char>`

6.866.2.9 `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMarshaledProperties () const [inline]`

Get the marshaledProperties field.

Returns

const reference to a `std::vector<char>`

6.866.2.10 `long long activemq::commands::WireFormatInfo::getMaxInactivityDuration () const`

Returns the currently configured Max Inactivity duration.

Returns

the set inactivity duration value.

6.866.2.11 `long long activemq::commands::WireFormatInfo::getMaxInactivityDurationInitialDelay () const`

Returns the currently configured Max Inactivity Initial Delay duration.

Returns

the set inactivity duration initial delay value.

6.866.2.12 `virtual util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties ()`
[inline, virtual]

Gets the Properties for this **Command** (p. 1107).

Returns

the Properties object for this **Command** (p. 1107).

6.866.2.13 `virtual const util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties ()`
`const` [inline, virtual]

Gets the Properties for this **Command** (p. 1107).

Returns

the Properties object for this **Command** (p. 1107).

6.866.2.14 `int activemq::commands::WireFormatInfo::getVersion () const`
[inline]

Get the current Wireformat Version.

Returns

int that identifies the version

6.866.2.15 `bool activemq::commands::WireFormatInfo::isCacheEnabled () const`

Checks if the cacheEnabled flag is on.

Returns

true if the flag is on.

6.866.2.16 `virtual bool activemq::commands::WireFormatInfo::isMarshalAware () const` [inline, virtual]

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

Returns

boolean indicating desire to be in marshaling stages

Reimplemented from **activemq::commands::BaseDataStructure** (p. 767).

6.866.2.17 **bool** **activemq::commands::WireFormatInfo::isSizePrefixDisabled** ()
 const

Checks if the sizePrefixDisabled flag is on.

Returns

true if the flag is on.

6.866.2.18 **bool** **activemq::commands::WireFormatInfo::isStackTraceEnabled** ()
 const

Checks if the stackTraceEnabled flag is on.

Returns

true if the flag is on.

6.866.2.19 **bool** **activemq::commands::WireFormatInfo::isTcpNoDelayEnabled** ()
 const

Checks if the tcpNoDelayEnabled flag is on.

Returns

true if the flag is on.

6.866.2.20 **bool** **activemq::commands::WireFormatInfo::isTightEncodingEnabled** ()
 const

Checks if the tightEncodingEnabled flag is on.

Returns

true if the flag is on.

6.866.2.21 **bool** **activemq::commands::WireFormatInfo::isValid** () **const**

Determines if we think this is a Valid **WireFormatInfo** (p. 3717) command.

Returns

true if its valid.

6.866.2.22 **virtual bool** **activemq::commands::WireFormatInfo::isWireFormatInfo** () **const** [inline, virtual]

Returns

answers true to the isWireFormatInfo query

Reimplemented from **activemq::commands::BaseCommand** (p. 700).

6.866.2.23 `void activemq::commands::WireFormatInfo::setCacheEnabled (bool cacheEnabled)`

Sets if the `cacheEnabled` flag is on.

Parameters

cacheEnabled - true to turn flag is on

6.866.2.24 `void activemq::commands::WireFormatInfo::setCacheSize (int value)`

Sets the Cache Size setting.

Parameters

value - value to set to the cache size.

6.866.2.25 `void activemq::commands::WireFormatInfo::setMagic (const std::vector< unsigned char > & magic) [inline]`

Sets the value of the magic field.

Parameters

magic - const std::vector<char>

6.866.2.26 `void activemq::commands::WireFormatInfo::setMarshaledProperties (const std::vector< unsigned char > & marshalledProperties) [inline]`

Sets the value of the `marshalledProperties` field.

Parameters

marshalledProperties The Byte Array vector that contains the marshaled form of the Message (p. 2358) properties, this is the data sent over the wire.

6.866.2.27 `void activemq::commands::WireFormatInfo::setMaxInactivityDuration (long long maxInactivityDuration)`

Sets the Max inactivity duration value.

Parameters

maxInactivityDuration - max time a client can be inactive.

6.866.2.28 `void activemq::commands::WireFormatInfo::setMaxInactivityDurationInitialDelay (long long maxInactivityDurationInitialDelay)`

Sets the Max inactivity initial delay duration value.

Parameters

maxInactivityDurationInitialDelay - time before the inactivity delay is checked.

6.866.2.29 `virtual void activemq::commands::WireFormatInfo::setProperties (const util::PrimitiveMap & map) [inline, virtual]`

Sets the Properties for this **Command** (p. 1107).

Parameters

map - PrimitiveMap to copy

6.866.2.30 `void activemq::commands::WireFormatInfo::setSizePrefixDisabled (bool sizePrefixDisabled)`

Sets if the sizePrefixDisabled flag is on.

Parameters

sizePrefixDisabled - true to turn flag is on

6.866.2.31 `void activemq::commands::WireFormatInfo::setStackTraceEnabled (bool stackTraceEnabled)`

Sets if the stackTraceEnabled flag is on.

Parameters

stackTraceEnabled - ture to turn flag is on

6.866.2.32 `void activemq::commands::WireFormatInfo::setTcpNoDelayEnabled (bool tcpNoDelayEnabled)`

Sets if the tcpNoDelayEnabled flag is on.

Parameters

tcpNoDelayEnabled - ture to turn flag is on

6.866.2.33 `void activemq::commands::WireFormatInfo::setTightEncodingEnabled (bool tightEncodingEnabled)`

Sets if the `tightEncodingEnabled` flag is on.

Parameters

tightEncodingEnabled - true to turn flag is on

6.866.2.34 `void activemq::commands::WireFormatInfo::setVersion (int version)`
[inline]

Set the current Wireformat Version.

Parameters

version - int that identifies the version

6.866.2.35 `virtual std::string activemq::commands::WireFormatInfo::toString ()`
`const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 700).

6.866.2.36 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::commands::WireFormatInfo::visit (`
`activemq::state::CommandVisitor * visitor) throw (`
`exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 3076) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p.1112).

6.866.3 Field Documentation

6.866.3.1 `const unsigned char activemq::commands::WireFormatInfo::ID_ - WIREFORMATINFO = 1` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/WireFormatInfo.h`

6.867 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3728).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.867.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3728). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.867.2 Constructor & Destructor Documentation

6.867.2.1 `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

6.867.2.2 `virtual activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

6.867.3 Member Function Documentation

6.867.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.867.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

6.867.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.867.3.4 virtual void `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.867.3.5 virtual int `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.867.3.6 virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1539).

6.867.3.7 virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h`

6.868 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for `WireFormatInfoMarshaller` (p. 3731).

#include <src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.868.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3731). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.868.2 Constructor & Destructor Documentation

6.868.2.1 `activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

6.868.2.2 `virtual activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

6.868.3 Member Function Documentation

6.868.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.868.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.868.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.868.3.4 virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.868.3.5 virtual int activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.868.3.6 virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.868.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h`

6.869 **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMar** **Class Reference**

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3735).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.869.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3735). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.869.2 Constructor & Destructor Documentation

6.869.2.1 `activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::WireFormatInfoMarshaller () [inline]`

6.869.2.2 `virtual activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller () [inline, virtual]`

6.869.3 Member Function Documentation

6.869.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.869.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.869.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.869.3.4 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1525).

6.869.3.5 virtual int activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1532).

6.869.3.6 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.869.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h`

6.870 **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMar** **Class Reference**

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3739).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller**:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.870.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3739). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.870.2 Constructor & Destructor Documentation

6.870.2.1 `activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::WireFormatInfoMarshaller () [inline]`

6.870.2.2 `virtual activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller () [inline, virtual]`

6.870.3 Member Function Documentation

6.870.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.870.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.870.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.870.3.4 virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.870.3.5 virtual int activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.870.3.6 virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.870.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h`

6.871 **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMar** Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3743).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.871.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3743). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.871.2 Constructor & Destructor Documentation

6.871.2.1 `activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::WireFormatInfoMarshaller () [inline]`

6.871.2.2 `virtual activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller () [inline, virtual]`

6.871.3 Member Function Documentation

6.871.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.871.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.871.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.871.3.4 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.871.3.5 virtual int activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.871.3.6 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.871.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h`

6.872 **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMar** Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3747).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.872.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3747). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.872.2 Constructor & Destructor Documentation

6.872.2.1 `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::WireFormatInfoMarshaller () [inline]`

6.872.2.2 `virtual activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller () [inline, virtual]`

6.872.3 Member Function Documentation

6.872.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1505).

6.872.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.872.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1518).

6.872.3.4 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1525).

6.872.3.5 virtual int activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1532).

6.872.3.6 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1539).

```
6.872.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1546).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**WireFormatInfoMarshaller.h**

6.873 activemq::wireformat::WireFormatNegotiator Class Reference

Defines a **WireFormatNegotiator** (p. 3751) which allows a **WireFormat** (p. 3712) to.

```
#include <src/main/activemq/wireformat/WireFormatNegotiator.h>
```

Inheritance diagram for **activemq::wireformat::WireFormatNegotiator**:

Public Member Functions

- **WireFormatNegotiator** (const **Pointer**< **transport::Transport** > &next)
Constructor.
- virtual ~**WireFormatNegotiator** ()

6.873.1 Detailed Description

Defines a **WireFormatNegotiator** (p. 3751) which allows a **WireFormat** (p. 3712) to.

6.873.2 Constructor & Destructor Documentation

- 6.873.2.1** **activemq::wireformat::WireFormatNegotiator::WireFormatNegotiator** (
const **Pointer**< **transport::Transport** > & *next*) [inline]

Constructor.

Parameters

next - the next Transport in the chain

- 6.873.2.2** virtual
activemq::wireformat::WireFormatNegotiator::~~WireFormatNegotiator (
) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatNegotiator.h**

6.874 activemq::wireformat::WireFormatRegistry Class Reference

Registry of all **WireFormat** (p. 3712) Factories that are available to the client at runtime.

```
#include <src/main/activemq/wireformat/WireFormatRegistry.h>
```

Public Member Functions

- virtual ~**WireFormatRegistry** ()
- **WireFormatFactory** * **findFactory** (const std::string &name) const throw (decaf::lang::exceptions::NoSuchElementException)
*Gets a Registered **WireFormatFactory** (p. 3716) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*
- void **registerFactory** (const std::string &name, **WireFormatFactory** *factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

*Registers a new **WireFormatFactory** (p. 3716) with this Registry.*

- void **unregisterFactory** (const std::string &name)
Unregisters the Factory with the given name and deletes that instance of the Factory.
- std::vector< std::string > **getWireFormatNames** () const
Retrieves a list of the names of all the Registered WireFormat's in this Registry.

Static Public Member Functions

- static **WireFormatRegistry** & **getInstance** ()
*Gets the single instance of the **WireFormatRegistry** (p. 3752).*

6.874.1 Detailed Description

Registry of all **WireFormat** (p. 3712) Factories that are available to the client at runtime. New WireFormat's must have a factory registered here before a connection attempt is made.

Since

3.0

6.874.2 Constructor & Destructor Documentation

- 6.874.2.1 **virtual**
activemq::wireformat::WireFormatRegistry::~~WireFormatRegistry ()
[virtual]

6.874.3 Member Function Documentation

- 6.874.3.1 **WireFormatFactory*** **activemq::wireformat::WireFormatRegistry::findFactory**
(const std::string & *name*) const throw (
decaf::lang::exceptions::NoSuchElementException)

Gets a Registered **WireFormatFactory** (p. 3716) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

Parameters

name The name of the Factory to find in the Registry.

Returns

the Factory registered under the given name.

Exceptions

NoSuchElementException if no factory is registered with that name.

6.874.3.2 `static WireFormatRegistry& activemq::wireformat::WireFormatRegistry::getInstance () [static]`

Gets the single instance of the **WireFormatRegistry** (p. 3752).

Returns

reference to the single instance of this Registry

6.874.3.3 `std::vector<std::string> activemq::wireformat::WireFormatRegistry::getWireFormatNames () const`

Retrieves a list of the names of all the Registered WireFormat's in this Registry.

Returns

stl vector of strings with all the **WireFormat** (p. 3712) names registered.

6.874.3.4 `void activemq::wireformat::WireFormatRegistry::registerFactory (const std::string & name, WireFormatFactory * factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)`

Registers a new **WireFormatFactory** (p. 3716) with this Registry.

If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

Parameters

name The name of the new Factory to register.

factory The new Factory to add to the Registry.

Exceptions

IllegalArgumentException is name is the empty string.

NullPointerException if the Factory is Null.

6.874.3.5 `void activemq::wireformat::WireFormatRegistry::unregisterFactory (const std::string & name)`

Unregisters the Factory with the given name and deletes that instance of the Factory.

Parameters

name Name of the Factory to unregister and destroy

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatRegistry.h**

6.875 activemq::transport::inactivity::WriteChecker Class Reference

Runnable class used by the {}.

```
#include <src/main/activemq/transport/inactivity/WriteChecker.h>
```

Inheritance diagram for `activemq::transport::inactivity::WriteChecker`:

Public Member Functions

- **WriteChecker** (**InactivityMonitor** *parent)
- virtual ~**WriteChecker** ()
- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

6.875.1 Detailed Description

Runnable class used by the {}.

See also

InactivityMonitor (p. 1874)} to make periodic writes to the underlying **transport** (p. 89) if no other write activity is going on in order to more quickly detect failures of the connection to the broker.

Since

3.1.0

6.875.2 Constructor & Destructor Documentation

6.875.2.1 `activemq::transport::inactivity::WriteChecker::WriteChecker (InactivityMonitor * parent)`

6.875.2.2 `virtual activemq::transport::inactivity::WriteChecker::~~WriteChecker () [virtual]`

6.875.3 Member Function Documentation

6.875.3.1 `virtual void activemq::transport::inactivity::WriteChecker::run () [virtual]`

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3112).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/inactivity/WriteChecker.h`

6.876 decaf::io::Writer Class Reference

```
#include <src/main/decaf/io/Writer.h>
```

Inheritance diagram for decaf::io::Writer:

Public Member Functions

- **Writer** ()
- virtual **~Writer** ()
- virtual void **write** (char v) throw (decaf::io::IOException)
Writes an single byte char value.
- virtual void **write** (const std::vector< char > &buffer) throw (decaf::io::IOException)
Writes an array of Chars.
- virtual void **write** (const char *buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
Writes a byte array to the output stream.
- virtual void **write** (const char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)
Writes a byte array to the output stream.
- virtual void **write** (const std::string &str) throw (decaf::io::IOException)
Writes a string.
- virtual void **write** (const std::string &str, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException)
Writes a string.
- virtual **decaf::lang::Appendable** & **append** (char value) throw (decaf::io::IOException)
Appends the specified character to this Appendable.
- virtual **decaf::lang::Appendable** & **append** (const **decaf::lang::CharSequence** *csq) throw (decaf::io::IOException)
Appends the specified character sequence to this Appendable.
- virtual **decaf::lang::Appendable** & **append** (const **decaf::lang::CharSequence** *csq, int start, int end) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException)
Appends a subsequence of the specified character sequence to this Appendable.

Protected Member Functions

- virtual void **doWriteArrayBounded** (const char *buffer, int size, int offset, int length)=0 throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)

Override this method to customize the functionality of the method write(char buffer, int size, int offset, int length).*

- virtual void **doWriteChar** (char v) throw (decaf::io::IOException)
- virtual void **doWriteVector** (const std::vector< char > &buffer) throw (decaf::io::IOException)
- virtual void **doWriteArray** (const char *buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
- virtual void **doWriteString** (const std::string &str) throw (decaf::io::IOException)
- virtual void **doWriteStringBounded** (const std::string &str, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual **decaf::lang::Appendable** & **doAppendChar** (char value) throw (decaf::io::IOException)
- virtual **decaf::lang::Appendable** & **doAppendCharSequence** (const **decaf::lang::CharSequence** *csq) throw (decaf::io::IOException)
- virtual **decaf::lang::Appendable** & **doAppendCharSequenceStartEnd** (const **decaf::lang::CharSequence** *csq, int start, int end) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException)

6.876.1 Constructor & Destructor Documentation

6.876.1.1 decaf::io::Writer::Writer ()

6.876.1.2 virtual decaf::io::Writer::~~Writer () [virtual]

6.876.2 Member Function Documentation

6.876.2.1 virtual decaf::lang::Appendable& decaf::io::Writer::append (char value) throw (decaf::io::IOException) [virtual]

Appends the specified character to this Appendable.

Parameters

value The character to append.

Returns

a Reference to this Appendable

Exceptions

Exception if an error occurs.

Implements **decaf::lang::Appendable** (p. 667).

6.876.2.2 `virtual decaf::lang::Appendable& decaf::io::Writer::append (const decaf::lang::CharSequence * csq) throw (decaf::io::IOException)`
[virtual]

Appends the specified character sequence to this Appendable.

Parameters

csq The character sequence from which a subsequence will be appended. If *csq* is NULL, then characters will be appended as if *csq* contained the string "null".

Returns

a Reference to this Appendable.

Exceptions

Exception if an error occurs.

Implements **decaf::lang::Appendable** (p. 668).

6.876.2.3 `virtual decaf::lang::Appendable& decaf::io::Writer::append (const decaf::lang::CharSequence * csq, int start, int end) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException)` [virtual]

Appends a subsequence of the specified character sequence to this Appendable.

Parameters

csq - The character sequence from which a subsequence will be appended. If *csq* is NULL, then characters will be appended as if *csq* contained the string "null".

start The index of the first character in the subsequence.

end The index of the character following the last character in the subsequence.

Returns

a Reference to this Appendable

Exceptions

Exception if an error occurs.

IndexOutOfBoundsException *start* is greater than *end*, or *end* is greater than *csq.length()*

Implements **decaf::lang::Appendable** (p. 667).

- 6.876.2.4 virtual `decaf::lang::Appendable& decaf::io::Writer::doAppendChar (char value) throw (decaf::io::IOException)` [protected, virtual]
- 6.876.2.5 virtual `decaf::lang::Appendable& decaf::io::Writer::doAppendCharSequence (const decaf::lang::CharSequence * csq) throw (decaf::io::IOException)` [protected, virtual]
- 6.876.2.6 virtual `decaf::lang::Appendable& decaf::io::Writer::doAppendCharSequenceStartEnd (const decaf::lang::CharSequence * csq, int start, int end) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException)` [protected, virtual]
- 6.876.2.7 virtual `void decaf::io::Writer::doWriteArray (const char * buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)` [protected, virtual]
- 6.876.2.8 virtual `void decaf::io::Writer::doWriteArrayBounded (const char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException)` [protected, pure virtual]

Override this method to customize the functionality of the method `write(char* buffer, int size, int offset, int length)`.

All subclasses must override this method to provide the basic **Writer** (p. 3756) functionality.

Implemented in **decaf::io::OutputStreamWriter** (p. 2727).

- 6.876.2.9 virtual `void decaf::io::Writer::doWriteChar (char v) throw (decaf::io::IOException)` [protected, virtual]
- 6.876.2.10 virtual `void decaf::io::Writer::doWriteString (const std::string & str) throw (decaf::io::IOException)` [protected, virtual]
- 6.876.2.11 virtual `void decaf::io::Writer::doWriteStringBounded (const std::string & str, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException)` [protected, virtual]
- 6.876.2.12 virtual `void decaf::io::Writer::doWriteVector (const std::vector< char > & buffer) throw (decaf::io::IOException)` [protected, virtual]
- 6.876.2.13 virtual `void decaf::io::Writer::write (char v) throw (decaf::io::IOException)` [virtual]

Writes an single byte char value.

Parameters

- v* The value to be written.

Exceptions

IOException (p. 2003) thrown if an error occurs.

6.876.2.14 virtual void decaf::io::Writer::write (const std::string & *str*) throw (decaf::io::IOException) [virtual]

Writes a string.

Parameters

str The string to be written.

Exceptions

IOException (p. 2003) thrown if an error occurs.

6.876.2.15 virtual void decaf::io::Writer::write (const std::string & *str*, int *offset*, int *length*) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes a string.

Parameters

str The string to be written.

offset The position in the array to start writing from.

length The number of bytes in the array to write.

Exceptions

IOException (p. 2003) thrown if an error occurs.

IndexOutOfBoundsException if offset+length is greater than the string length.

6.876.2.16 virtual void decaf::io::Writer::write (const char * *buffer*, int *size*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException) [virtual]

Writes a byte array to the output stream.

Parameters

buffer The byte array to write (cannot be NULL).

size The size in bytes of the buffer passed.

Exceptions

IOException (p. 2003) if an I/O error occurs.

NullPointerException if buffer is NULL.

6.876.2.17 `virtual void decaf::io::Writer::write (const std::vector< char > & buffer) throw (decaf::io::IOException) [virtual]`

Writes an array of Chars.

Parameters

buffer The array to be written.

Exceptions

IOException (p. 2003) thrown if an error occurs.

6.876.2.18 `virtual void decaf::io::Writer::write (const char * buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes a byte array to the output stream.

Parameters

buffer The byte array to write (cannot be NULL).

size The size in bytes of the buffer passed.

offset The position in the array to start writing from.

length The number of bytes in the array to write.

Exceptions

IOException (p. 2003) if an I/O error occurs.

NullPointerException if buffer is NULL.

IndexOutOfBoundsException if offset + length > size of the buffer.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Writer.h**

6.877 decaf::security::auth::x500::X500Principal Class Reference

```
#include <src/main/decaf/security/auth/x500/X500Principal.h>
```

Inheritance diagram for decaf::security::auth::x500::X500Principal:

Public Member Functions

- virtual `~X500Principal ()`
- virtual `std::string getName () const =0`

Provides the name of this principal.

- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0
- virtual int **hashCode** () const =0

6.877.1 Constructor & Destructor Documentation

6.877.1.1 virtual decaf::security::auth::x500::X500Principal::~X500Principal ()
[inline, virtual]

6.877.2 Member Function Documentation

6.877.2.1 virtual void decaf::security::auth::x500::X500Principal::getEncoded (std::vector< unsigned char > & *output*) const [pure virtual]

6.877.2.2 virtual std::string decaf::security::auth::x500::X500Principal::getName () const [pure virtual]

Provides the name of this principal.

Returns

the name of this principal.

Implements **decaf::security::Principal** (p. 2831).

6.877.2.3 virtual int decaf::security::auth::x500::X500Principal::hashCode () const [pure virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/security/auth/x500/X500Principal.h

6.878 decaf::security::cert::X509Certificate Class Reference

Base interface for all identity certificates.

```
#include <src/main/decaf/security/cert/X509Certificate.h>
```

Inheritance diagram for decaf::security::cert::X509Certificate:

Public Member Functions

- virtual ~X509Certificate ()
- virtual void **checkValidity** () const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual void **checkValidity** (const decaf::util::Date &date) const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual int **getBasicConstraints** () const =0

- virtual void **getIssuerUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getIssuerX500Principal** () const =0
- virtual void **getKeyUsage** (std::vector< unsigned char > &output) const =0
- virtual Date **getNotAfter** () const =0
- virtual Date **getNotBefore** () const =0
- virtual std::string **getSigAlgName** () const =0
- virtual std::string **getSigAlgOID** () const =0
- virtual void **getSigAlgParams** (std::vector< unsigned char > &output) const =0
- virtual void **getSignature** (std::vector< unsigned char > &output) const =0
- virtual void **getSubjectUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getSubjectX500Principal** () const =0
- virtual void **getTBSCertificate** (std::vector< unsigned char > &output) const =0 throw
(CertificateEncodingException)
- virtual int **getVersion** () const =0

6.878.1 Detailed Description

Base interface for all identity certificates.

6.878.2 Constructor & Destructor Documentation

6.878.2.1 virtual decaf::security::cert::X509Certificate::~X509Certificate ()
[inline, virtual]

6.878.3 Member Function Documentation

6.878.3.1 virtual void decaf::security::cert::X509Certificate::checkValidity () const
throw (CertificateExpiredException, CertificateNotYetValidException)
[pure virtual]

6.878.3.2 virtual void decaf::security::cert::X509Certificate::checkValidity (const
decaf::util::Date & *date*) const throw (CertificateExpiredException,
CertificateNotYetValidException) [pure virtual]

6.878.3.3 virtual int decaf::security::cert::X509Certificate::getBasicConstraints () const
[pure virtual]

6.878.3.4 virtual void decaf::security::cert::X509Certificate::getIssuerUniqueID (
std::vector< bool > & *output*) const [pure virtual]

6.878.3.5 virtual const X500Principal* de-
caf::security::cert::X509Certificate::getIssuerX500Principal
() const [pure virtual]

6.878.3.6 virtual void decaf::security::cert::X509Certificate::getKeyUsage (
std::vector< unsigned char > & *output*) const [pure virtual]

6.878.3.7 virtual Date decaf::security::cert::X509Certificate::getNotAfter ()
const [pure virtual]

6.878.3.8 virtual Date decaf::security::cert::X509Certificate::getNotBefore ()
const [pure virtual]

6.878.3.9 virtual std::string decaf::security::cert::X509Certificate::getSigAlgName () const
[pure virtual]

6.878.3.10 virtual std::string decaf::security::cert::X509Certificate::getSigAlgOID () const
[pure virtual]

6.878.3.11 virtual void decaf::security::cert::X509Certificate::getSigAlgParams (
std::vector< unsigned char > & *output*) const [pure virtual]

6.878.3.12 virtual void decaf::security::cert::X509Certificate::getSignature (
std::vector< unsigned char > & *output*) const [pure virtual]

6.878.3.13 virtual void decaf::security::cert::X509Certificate::getSubjectUniqueID (
std::vector< bool > & *output*) const [pure virtual]

6.878.3.14 virtual const X500Principal* de-
caf::security::cert::X509Certificate::getSubjectX500Principal () const
[pure virtual]

6.878.3.15 virtual void decaf::security::cert::X509Certificate::getTBSCertificate
(std::vector< unsigned char > & *output*) const throw (
CertificateEncodingException) [pure virtual]

6.878.3.16 virtual int decaf::security::cert::X509Certificate::getVersion () const
[pure virtual]

Generated on Sun Sep 11 2011 18:21:35 for activeq-opp-3.2.1 by Doxygen

- `src/main/decaf/security/cert/X509Certificate.h`

6.879 `activemq::commands::XATransactionId` Class Reference

```
#include <src/main/activemq/commands/XATransactionId.h>
```

Inheritance diagram for `activemq::commands::XATransactionId`:

Public Types

- `typedef decaf::lang::PointerComparator< XATransactionId > COMPARATOR`

Public Member Functions

- `XATransactionId ()`
- `XATransactionId (const XATransactionId &other)`
- `virtual ~XATransactionId ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaller share.
- `virtual XATransactionId * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- `virtual void copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this object's members, overwriting any existing data.
- `virtual std::string toString () const`
*Returns a string containing the information for this **DataStructure** (p. 1553) such as its type and value of its elements.*
- `virtual bool equals (const DataStructure *value) const`
*Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent.*
- `virtual int getFormatId () const`
- `virtual void setFormatId (int formatId)`
- `virtual const std::vector< unsigned char > & getGlobalTransactionId () const`
- `virtual std::vector< unsigned char > & getGlobalTransactionId ()`
- `virtual void setGlobalTransactionId (const std::vector< unsigned char > &globalTransactionId)`
- `virtual const std::vector< unsigned char > & getBranchQualifier () const`
- `virtual std::vector< unsigned char > & getBranchQualifier ()`
- `virtual void setBranchQualifier (const std::vector< unsigned char > &branchQualifier)`
- `virtual int compareTo (const XATransactionId &value) const`
- `virtual bool equals (const XATransactionId &value) const`
- `virtual bool operator== (const XATransactionId &value) const`
- `virtual bool operator< (const XATransactionId &value) const`
- `XATransactionId & operator= (const XATransactionId &other)`

Static Public Attributes

- static const unsigned char **ID_XATRANSACTIONID** = 112

Protected Attributes

- int **formatId**
- std::vector< unsigned char > **globalTransactionId**
- std::vector< unsigned char > **branchQualifier**

6.879.1 Member Typedef Documentation

6.879.1.1 typedef decaf::lang::PointerComparator<XATransactionId>
 activemq::commands::XATransactionId::COMPARATOR

6.879.2 Constructor & Destructor Documentation

6.879.2.1 activemq::commands::XATransactionId::XATransactionId ()

6.879.2.2 activemq::commands::XATransactionId::XATransactionId (const
 XATransactionId & *other*)

6.879.2.3 virtual activemq::commands::XATransactionId::~~XATransactionId ()
 [virtual]

6.879.3 Member Function Documentation

6.879.3.1 virtual XATransactionId* ac-
 tivemq::commands::XATransactionId::cloneDataStructure
 () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.879.3.2 virtual int activemq::commands::XATransactionId::compareTo (const
 XATransactionId & *value*) const [virtual]

6.879.3.3 virtual void activemq::commands::XATransactionId::copyDataStructure (const
 DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.879.3.4 `virtual bool activemq::commands::XATransactionId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1553) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.879.3.5 `virtual bool activemq::commands::XATransactionId::equals (const XATransactionId & value) const` [virtual]

6.879.3.6 `virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier () const` [virtual]

6.879.3.7 `virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier ()` [virtual]

6.879.3.8 `virtual unsigned char activemq::commands::XATransactionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1553) type copy.

- 6.879.3.9 virtual int activemq::commands::XATransactionId::getFormatId ()
 const [virtual]

- 6.879.3.10 virtual const std::vector<unsigned char>& ac-
 tivemq::commands::XATransactionId::getGlobalTransactionId ()
 const [virtual]

- 6.879.3.11 virtual std::vector<unsigned char>& ac-
 tivemq::commands::XATransactionId::getGlobalTransactionId ()
 [virtual]

- 6.879.3.12 virtual bool activemq::commands::XATransactionId::operator< (const
 XATransactionId & *value*) const [virtual]

- 6.879.3.13 XATransactionId& activemq::commands::XATransactionId::operator= (const
 XATransactionId & *other*)

- 6.879.3.14 virtual bool activemq::commands::XATransactionId::operator== (const
 XATransactionId & *value*) const [virtual]

- 6.879.3.15 virtual void activemq::commands::XATransactionId::setBranchQualifier
 (const std::vector< unsigned char > & *branchQualifier*) [virtual]

- 6.879.3.16 virtual void activemq::commands::XATransactionId::setFormatId (int
 formatId) [virtual]

- 6.879.3.17 virtual void ac-
 tivemq::commands::XATransactionId::setGlobalTransactionId (const
 std::vector< unsigned char > & *globalTransactionId*) [virtual]

- 6.879.3.18 virtual std::string activemq::commands::XATransactionId::toString ()
 const [virtual]

Returns a string containing the information for this **DataSet** (p.1553) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.879.4 Field Documentation

- 6.879.4.1 `std::vector<unsigned char> activemq::commands::XATransactionId::branchQualifier` [protected]
- 6.879.4.2 `int activemq::commands::XATransactionId::formatId` [protected]
- 6.879.4.3 `std::vector<unsigned char> activemq::commands::XATransactionId::globalTransactionId` [protected]
- 6.879.4.4 `const unsigned char activemq::commands::XATransactionId::ID_ - XATRANSACTIONID = 112` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/XATransactionId.h`

6.880 `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `XATransactionIdMarshaller` (p. 3769).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller`:

Public Member Functions

- `XATransactionIdMarshaller ()`
- `virtual ~XATransactionIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.880.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3769). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.880.2 Constructor & Destructor Documentation

6.880.2.1 **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::XATransactionIdMarshaller** () [inline]

6.880.2.2 **virtual**
activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::~~XATransactionIdMarshaller () [inline, virtual]

6.880.3 Member Function Documentation

6.880.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.880.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1511).

6.880.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3592).

6.880.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3592).

6.880.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3593).

6.880.3.6 virtual void `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightMarshal2`
(`OpenWireFormat` * *wireFormat*, `commands::DataStructure`
* *dataStructure*, `decaf::io::DataOutputStream` * *dataOut*,
`utils::BooleanStream` * *bs*) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3593).

6.880.3.7 virtual void `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightUnmarshal`
(`OpenWireFormat` * *wireFormat*, `commands::DataStructure`
* *dataStructure*, `decaf::io::DataInputStream` * *dataIn*,
`utils::BooleanStream` * *bs*) throw (`decaf::io::IOException`) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3594).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h`

6.881 **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3773).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.881.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3773). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.881.2 Constructor & Destructor Documentation

6.881.2.1 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::XATransactionIdMarshaller () [inline]

6.881.2.2 virtual activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::~~XATransactionIdMarshaller () [inline, virtual]

6.881.3 Member Function Documentation

6.881.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1505).

6.881.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1511).

```

6.881.3.3 virtual void ac-
tivismq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3581).

```

6.881.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3581).

```

6.881.3.5 virtual int ac-
tivismq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3582).

```
6.881.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3582).

```
6.881.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3583).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h`

6.882 **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3777).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller**:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.882.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p.3777). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.882.2 Constructor & Destructor Documentation

6.882.2.1 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::XATransactionIdMarshaller () [inline]

6.882.2.2 virtual
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::~XATransactionIdMarshaller () [inline, virtual]

6.882.3 Member Function Documentation

6.882.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.882.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.882.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3588).

6.882.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3589).

6.882.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3589).

```
6.882.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3590).

```
6.882.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3590).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h`

6.883 **activemq::wireformat::openwire::marshal::v1::XATransactionIdMa** Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3781).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller**:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.883.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p.3781). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.883.2 Constructor & Destructor Documentation

6.883.2.1 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::XATransactionIdMarshaller () [inline]

6.883.2.2 virtual activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::~~XATransactionIdMarshaller () [inline, virtual]

6.883.3 Member Function Documentation

6.883.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.883.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.883.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3577).

6.883.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3578).

6.883.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3578).

```
6.883.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3579).

```
6.883.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3579).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h`

6.884 **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3785).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller**:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.884.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p.3785). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.884.2 Constructor & Destructor Documentation

6.884.2.1 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::XATransactionIdMarshaller () [inline]

6.884.2.2 virtual activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::~~XATransactionIdMarshaller () [inline, virtual]

6.884.3 Member Function Documentation

6.884.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.884.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.884.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3585).

6.884.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3585).

6.884.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3586).

```
6.884.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3586).

```
6.884.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3587).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h`

6.885 **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3789).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller**:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.885.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p.3789). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.885.2 Constructor & Destructor Documentation

6.885.2.1 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::XATransactionIdMarshaller () [inline]

6.885.2.2 virtual activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::~XATransactionIdMarshaller () [inline, virtual]

6.885.3 Member Function Documentation

6.885.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1505).

6.885.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1511).

6.885.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3574).

6.885.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3574).

6.885.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3575).

```
6.885.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3575).

```
6.885.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3576).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h`

6.886 decaf::util::logging::XMLFormatter Class Reference

Format a **LogRecord** (p. 2261) into a standard XML format.

```
#include <src/main/decaf/util/logging/XMLFormatter.h>
```

Inheritance diagram for `decaf::util::logging::XMLFormatter`:

Public Member Functions

- **XMLFormatter** ()
- virtual **~XMLFormatter** ()
- virtual `std::string format` (const **LogRecord** &record) const
*Converts a **LogRecord** (p. 2261) into an XML string.*
- virtual `std::string getHead` (const **Handler** *handler)
Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.
- virtual `std::string getTail` (const **Handler** *handler)
Returns the tail string for a set of log records formatted as XML strings.

6.886.1 Detailed Description

Format a **LogRecord** (p. 2261) into a standard XML format. TODO - Currently only outputs UTF-8 The **XMLFormatter** (p. 3793) can be used with arbitrary character encodings, but it is recommended that it normally be used with UTF-8. The character encoding can be set on the output **Handler** (p. 1852).

Since

1.0

6.886.2 Constructor & Destructor Documentation

6.886.2.1 `decaf::util::logging::XMLFormatter::XMLFormatter ()`

6.886.2.2 `virtual decaf::util::logging::XMLFormatter::~~XMLFormatter ()`
[virtual]

6.886.3 Member Function Documentation

6.886.3.1 `virtual std::string decaf::util::logging::XMLFormatter::format (const LogRecord & record) const` [virtual]

Converts a **LogRecord** (p. 2261) into an XML string.

Parameters

record The log record to be formatted.

Returns

the log record formatted as an XML string.

Implements **decaf::util::logging::Formatter** (p. 1839).

6.886.3.2 `virtual std::string decaf::util::logging::XMLFormatter::getHead (const Handler * handler)` [virtual]

Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.

Parameters

handler The output handler, may be NULL.

Returns

the header string for log records formatted as XML strings.

6.886.3.3 `virtual std::string decaf::util::logging::XMLFormatter::getTail (const Handler * handler)` [virtual]

Returns the tail string for a set of log records formatted as XML strings.

Parameters

handler The output handler, may be NULL.

Returns

the tail string for log records formatted as XML strings.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/XMLFormatter.h`

6.887 z_stream_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/zlib.h>
```

Data Fields

- `Bytef * next_in`
- `uInt avail_in`
- `uLong total_in`
- `Bytef * next_out`
- `uInt avail_out`
- `uLong total_out`
- `char * msg`
- `struct internal_state FAR * state`
- `alloc_func zalloc`
- `free_func zfree`
- `voidpf opaque`
- `int data_type`
- `uLong Adler`
- `uLong reserved`

6.887.1 Field Documentation

6.887.1.1 `uLong z_stream_s::Adler`

6.887.1.2 `uInt z_stream_s::avail_in`

6.887.1.3 `uInt z_stream_s::avail_out`

6.887.1.4 `int z_stream_s::data_type`

6.887.1.5 `char* z_stream_s::msg`

6.887.1.6 `Bytef* z_stream_s::next_in`

6.887.1.7 `Bytef* z_stream_s::next_out`

6.887.1.8 `voidpf z_stream_s::opaque`

6.887.1.9 `uLong z_stream_s::reserved`

6.887.1.10 `struct internal_state FAR* z_stream_s::state`

6.887.1.11 `uLong z_stream_s::total_in`

6.887.1.12 `uLong z_stream_s::total_out`

6.887.1.13 `alloc_func z_stream_s::zalloc`

6.887.1.14 `free_func z_stream_s::zfree`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/zlib.h`

6.888 decaf::util::zip::ZipException Class Reference

```
#include <src/main/decaf/util/zip/ZipException.h>
```

Inheritance diagram for `decaf::util::zip::ZipException`:

Public Member Functions

- **ZipException** () throw ()
Default Constructor.
- **ZipException** (const **lang::Exception** &ex) throw ()
Copy Constructor.
- **ZipException** (const **ZipException** &ex) throw ()
Copy Constructor.
- **ZipException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ZipException** (const std::exception *cause) throw ()
Constructor.
- **ZipException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **ZipException** * clone () const
Clones this exception.
- virtual ~**ZipException** () throw ()

6.888.1 Constructor & Destructor Documentation

6.888.1.1 decaf::util::zip::ZipException::ZipException () throw () [inline]

Default Constructor.

6.888.1.2 decaf::util::zip::ZipException::ZipException (const lang::Exception &ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.888.1.3 `decaf::util::zip::ZipException::ZipException (const ZipException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.888.1.4 `decaf::util::zip::ZipException::ZipException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.888.1.5 `decaf::util::zip::ZipException::ZipException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.888.1.6 `decaf::util::zip::ZipException::ZipException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.888.1.7 virtual decaf::util::zip::ZipException::~ZipException () throw ()
[inline, virtual]

6.888.2 Member Function Documentation

6.888.2.1 virtual ZipException* decaf::util::zip::ZipException::clone () const
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2005).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**ZipException.h**

Chapter 7

File Documentation

7.1 src/main/activemq/cmsutil/CachedConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::CachedConsumer**
A cached message consumer contained within a pooled session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.2 src/main/activemq/cmsutil/CachedProducer.h File Reference

```
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::CachedProducer**
A cached message producer contained within a pooled session.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.3 src/main/activemq/cmsutil/CmsAccessor.h File Reference

```
#include <cms/ConnectionFactory.h>
#include <activemq/cmsutil/ResourceLifecycleManager.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **activemq::cmsutil::CmsAccessor**

*Base class for **activemq.cmsutil.CmsTemplate** (p. 1083) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1228) to operate on.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.4 src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsAccessor.h>
#include <activemq/cmsutil/DynamicDestinationResolver.h>
```

Data Structures

- class **activemq::cmsutil::CmsDestinationAccessor**

*Extends the **CmsAccessor** (p. 1068) to add support for resolving destination names.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.5 src/main/activemq/cmsutil/CmsTemplate.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsDestinationAccessor.h>
#include <activemq/cmsutil/SessionCallback.h>
#include <activemq/cmsutil/ProducerCallback.h>
#include <activemq/cmsutil/SessionPool.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <cms/ConnectionFactory.h>
#include <cms/DeliveryMode.h>
#include <string>
```

Data Structures

- class **activemq::cmsutil::CmsTemplate**

CmsTemplate (p. 1083) simplifies performing synchronous CMS operations.

- class **activemq::cmsutil::CmsTemplate::ProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::SendExecutor**
- class **activemq::cmsutil::CmsTemplate::ReceiveExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.6 src/main/activemq/cmsutil/DestinationResolver.h File Reference

```
#include <cms/Session.h>
```

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::DestinationResolver**
*Resolves a CMS destination name to a **Destination**.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.7 src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference

```
#include <activemq/cmsutil/DestinationResolver.h>  
#include <cms/Session.h>  
#include <decaf/util/StlMap.h>  
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::DynamicDestinationResolver**
*Resolves a CMS destination name to a **Destination**.*
- class **activemq::cmsutil::DynamicDestinationResolver::SessionResolver**
Manages maps of names to topics and queues for a single session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.8 src/main/activemq/cmsutil/MessageCreator.h File Reference

```
#include <cms/Session.h>
```



```
#include <cms/Message.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::MessageCreator**
Creates the user-defined message to be sent by the `CmsTemplate` (p. 1083).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.9 src/main/activemq/cmsutil/PooledSession.h File Reference

```
#include <cms/Session.h>
#include <decaf/util/StlMap.h>
#include <activemq/cmsutil/CachedProducer.h>
#include <activemq/cmsutil/CachedConsumer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::PooledSession**
A pooled session object that wraps around a delegate session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.10 src/main/activemq/cmsutil/ProducerCallback.h File Reference

```
#include <cms/Session.h>
```

```
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::ProducerCallback**
Callback for sending a message to a CMS destination.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.11 src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference

```
#include <cms/Connection.h>
#include <cms/Session.h>
#include <cms/Destination.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
#include <decaf/util/StlList.h>
```

Data Structures

- class **activemq::cmsutil::ResourceLifecycleManager**
Manages the lifecycle of a set of CMS resources.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.12 src/main/decaf/internal/util/ResourceLifecycleManager.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/StlSet.h>
#include <decaf/internal/util/Resource.h>
```

Data Structures

- class **decaf::internal::util::ResourceLifecycleManager**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.13 src/main/activemq/cmsutil/SessionCallback.h File Reference

```
#include <cms/Session.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::SessionCallback**
Callback for executing any number of operations on a provided CMS Session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.14 src/main/activemq/cmsutil/SessionPool.h File Reference

```
#include <activemq/cmsutil/PooledSession.h>
```

```
#include <decaf/util/concurrent/Mutex.h>
#include <cms/Connection.h>
#include <list>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::SessionPool**
A pool of CMS sessions from the same connection and with the same acknowledge mode.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.15 src/main/activemq/commands/ActiveMQBlobMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/Message.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQBlobMessage**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.16 src/main/activemq/commands/ActiveMQBytesMessage.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <cms/BytesMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQBytesMessage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.17 src/main/activemq/commands/ActiveMQDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/ActiveMQProperties.h>
#include <cms/Destination.h>
#include <decaf/lang/Pointer.h>
#include <vector>
#include <string>
#include <map>
```

Data Structures

- class **activemq::commands::ActiveMQDestination**
- struct **activemq::commands::ActiveMQDestination::DestinationFilter**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.18 src/main/activemq/commands/ActiveMQMapMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <cms/MapMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQMapMessage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.19 src/main/activemq/commands/ActiveMQMessage.h File Reference

```
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
```

Data Structures

- class **activemq::commands::ActiveMQMessage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.20 src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference

```
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/core/ActiveMQConnection.h>
#include <activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <activemq/util/CMSExceptionSupport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <cms/IllegalStateException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWritableException.h>
```

Data Structures

- class **activemq::commands::ActiveMQMessageTemplate< T >**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.21 src/main/activemq/commands/ActiveMQObjectMessage.h File Reference

```
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/ObjectMessage.h>
#include <activemq/util/Config.h>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQObjectMessage`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.22 `src/main/activemq/commands/ActiveMQQueue.h` File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Queue.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQQueue`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.23 `src/main/activemq/commands/ActiveMQStreamMessage.h` File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessage.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/StreamMessage.h>
```



```
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQStreamMessage`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.24 src/main/activemq/commands/ActiveMQTempDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <cms/Closeable.h>
#include <vector>
#include <string>
```

Data Structures

- class `activemq::commands::ActiveMQTempDestination`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**
- namespace **activemq::commands**

7.25 src/main/activemq/commands/ActiveMQTempQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryQueue.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTempQueue**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.26 src/main/activemq/commands/ActiveMQTempTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryTopic.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTempTopic**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.27 src/main/activemq/commands/ActiveMQTextMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/TextMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTextMessage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.28 src/main/activemq/commands/ActiveMQTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Topic.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQTopic`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.29 `src/main/activemq/commands/BaseCommand.h` File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
```

Data Structures

- class `activemq::commands::BaseCommand`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.30 `src/main/activemq/commands/BaseDataStructure.h` File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <string>
#include <sstream>
```

Data Structures

- class `activemq::commands::BaseDataStructure`

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::commands**

7.31 src/main/activemq/commands/BooleanExpression.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::commands::BooleanExpression**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.32 src/main/activemq/commands/BrokerError.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::BrokerError**

This class represents an Exception sent from the Broker.

- struct **activemq::commands::BrokerError::StackTraceElement**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.33 src/main/activemq/commands/BrokerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::BrokerId**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.34 src/main/activemq/commands/BrokerInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::BrokerInfo`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.35 src/main/activemq/commands/Command.h File Reference

```
#include <string>
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class `activemq::commands::Command`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::state`
- namespace `activemq::commands`

7.36 src/main/activemq/commands/ConnectionControl.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConnectionControl`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.37 `src/main/activemq/commands/ConnectionError.h` File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConnectionError`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.38 `src/main/activemq/commands/ConnectionId.h` File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
```



```
#include <vector>
```

Data Structures

- class `activemq::commands::ConnectionId`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.39 src/main/activemq/commands/ConnectionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConnectionInfo`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.40 src/main/activemq/commands/ConsumerControl.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
```

```
#include <activemq/commands/ConsumerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConsumerControl`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.41 `src/main/activemq/commands/ConsumerId.h` File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConsumerId`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.42 src/main/activemq/commands/ConsumerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BooleanExpression.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConsumerInfo`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.43 src/main/activemq/commands/ControlCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ControlCommand`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.44 src/main/activemq/commands/DataArrayResponse.h File Reference

```
#include <activemq/commands/DataSet.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::DataArrayResponse**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.45 src/main/activemq/commands/DataResponse.h File Reference

```
#include <activemq/commands/DataSet.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::DataResponse**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.46 src/main/activemq/commands/DataSetructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/MarshalAware.h>
```

Data Structures

- class **activemq::commands::DataSetructure**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.47 src/main/activemq/commands/DestinationInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::DestinationInfo**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.48 src/main/activemq/commands/DiscoveryEvent.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::DiscoveryEvent**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.49 src/main/activemq/commands/ExceptionResponse.h File Reference

```
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ExceptionResponse**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.50 src/main/activemq/commands/FlushCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::FlushCommand**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.51 src/main/activemq/commands/IntegerResponse.h File Reference

```
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::IntegerResponse**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.52 src/main/activemq/commands/JournalQueueAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::JournalQueueAck**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.53 src/main/activemq/commands/JournalTopicAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```


Data Structures

- class `activemq::commands::JournalTopicAck`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.54 src/main/activemq/commands/JournalTrace.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::JournalTrace`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.55 src/main/activemq/commands/JournalTransaction.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::JournalTransaction`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.56 `src/main/activemq/commands/KeepAliveInfo.h` File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::KeepAliveInfo`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.57 `src/main/activemq/commands/LastPartialCommand.h` File Reference

```
#include <activemq/commands/PartialCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::LastPartialCommand`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.58 src/main/activemq/commands/LocalTransactionId.h File Reference

```
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::LocalTransactionId`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.59 src/main/activemq/commands/Message.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::Message**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**
- namespace **activemq::commands**

7.60 src/main/cms/Message.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <cms/CMSException.h>
#include <cms/IllegalStateException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class **cms::Message**
Root of all messages.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.61 src/main/activemq/commands/MessageAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::MessageAck**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.62 src/main/activemq/commands/MessageDispatch.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::MessageDispatch`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.63 `src/main/activemq/commands/MessageDispatchNotification.h` File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::MessageDispatchNotification`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.64 `src/main/activemq/commands/MessageId.h` File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/util/Config.h>
```

```
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::MessageId`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.65 src/main/activemq/commands/MessagePull.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::MessagePull`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.66 src/main/activemq/commands/NetworkBridgeFilter.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::NetworkBridgeFilter`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.67 src/main/activemq/commands/PartialCommand.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::PartialCommand`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.68 src/main/activemq/commands/ProducerAck.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ProducerAck`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.69 src/main/activemq/commands/ProducerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ProducerId`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.70 src/main/activemq/commands/ProducerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ProducerInfo**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.71 src/main/activemq/commands/RemoveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::RemoveInfo**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.72 src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::RemoveSubscriptionInfo**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.73 src/main/activemq/commands/ReplayCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ReplayCommand**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.74 src/main/activemq/commands/Response.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::Response**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.75 src/main/activemq/commands/SessionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::SessionId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.76 src/main/activemq/commands/SessionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::SessionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.77 src/main/activemq/commands/ShutdownInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ShutdownInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.78 src/main/activemq/commands/SubscriptionInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::SubscriptionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.79 src/main/activemq/commands/TransactionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::TransactionId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.80 src/main/activemq/commands/TransactionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::TransactionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.81 src/main/activemq/commands/WireFormatInfo.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <vector>
```

Data Structures

- class **activemq::commands::WireFormatInfo**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.82 src/main/activemq/commands/XATransactionId.h File Reference

```
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::XATransactionId**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.83 src/main/activemq/core/ActiveMQAckHandler.h File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::core::ActiveMQAckHandler**

Interface class that is used to give CMS Messages an interface to Ack themselves with.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**
- namespace **activemq::core**

7.84 src/main/activemq/core/ActiveMQConnection.h File Reference

```
#include <cms/Connection.h>
#include <activemq/util/Config.h>
#include <activemq/core/ActiveMQConnectionMetaData.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQConnection**

Concrete connection used for all connectors to the ActiveMQ broker.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.85 src/main/activemq/core/ActiveMQConnectionFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionFactory.h>
#include <cms/Connection.h>
```

Data Structures

- class **activemq::core::ActiveMQConnectionFactory**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.86 src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionMetaData.h>
```

Data Structures

- class **activemq::core::ActiveMQConnectionMetaData**

*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 233) class.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.87 src/main/activemq/core/ActiveMQConstants.h File Reference

```
#include <string>
#include <map>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::core::ActiveMQConstants**

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.

- class **activemq::core::ActiveMQConstants::StaticInitializer**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.88 src/main/activemq/core/ActiveMQConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/core/RedeliveryPolicy.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
```

```
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

Data Structures

- class `activemq::core::ActiveMQConsumer`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::core`

7.89 `src/main/activemq/core/ActiveMQProducer.h` File Reference

```
#include <cms/MessageProducer.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/util/MemoryUsage.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <memory>
```

Data Structures

- class `activemq::core::ActiveMQProducer`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::core`

7.90 src/main/activemq/core/ActiveMQQueueBrowser.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/MessageEnumeration.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
```

Data Structures

- class `activemq::core::ActiveMQQueueBrowser`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::core`

7.91 src/main/activemq/core/ActiveMQSession.h File Reference

```
#include <cms/Session.h>
#include <cms/ExceptionListener.h>
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/core/ActiveMQTransactionContext.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
```

```
#include <activemq/commands/TransactionId.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/Properties.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQSession**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.92 src/main/activemq/core/ActiveMQSessionExecutor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/threads/Task.h>
#include <activemq/threads/TaskRunner.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::ActiveMQSessionExecutor**

Delegate dispatcher for a single session.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.93 src/main/activemq/core/ActiveMQTransactionContext.h File Reference

```
#include <memory>
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/core/Synchronization.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::core::ActiveMQTransactionContext**

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.94 src/main/activemq/core/DispatchData.h File Reference

```
#include <stdlib.h>
#include <memory>
```

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::DispatchData**

Simple POJO that contains the information necessary to route a message to a specified consumer.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.95 src/main/activemq/core/Dispatcher.h File Reference

```
#include <activemq/commands/MessageDispatch.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::Dispatcher**

Interface for an object responsible for dispatching messages to consumers.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.96 src/main/activemq/core/MessageDispatchChannel.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/MessageDispatch.h>
```



```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class `activemq::core::MessageDispatchChannel`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::core`

7.97 src/main/activemq/core/policies/DefaultPrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/PrefetchPolicy.h>
```

Data Structures

- class `activemq::core::policies::DefaultPrefetchPolicy`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::core`
- namespace `activemq::core::policies`

7.98 src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/RedeliveryPolicy.h>
```

Data Structures

- class `activemq::core::policies::DefaultRedeliveryPolicy`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::core`
- namespace `activemq::core::policies`

7.99 src/main/activemq/core/PrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class `activemq::core::PrefetchPolicy`
Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::core`

7.100 src/main/activemq/core/RedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class `activemq::core::RedeliveryPolicy`
*Interface for a **RedeliveryPolicy** (p. 2972) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.101 src/main/activemq/core/Synchronization.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::core::Synchronization**

*Transacted Object **Synchronization** (p. 3477), used to sync the events of a Transaction with the items in the Transaction.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.102 src/main/activemq/exceptions/ActiveMQException.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <cms/CMSException.h>
```

```
#include <decaf/lang/Exception.h>
```

```
#include <activemq/exceptions/ExceptionDefines.h>
```

```
#include <stdarg.h>
```

```
#include <sstream>
```

Data Structures

- class **activemq::exceptions::ActiveMQException**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::exceptions**

7.103 src/main/activemq/exceptions/BrokerException.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/BrokerError.h>
#include <sstream>
```

Data Structures

- class **activemq::exceptions::BrokerException**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::exceptions**

7.104 src/main/activemq/exceptions/ExceptionDefines.h File Reference

Defines

- **#define AMQ_CATCH_RETHROW(type)**
Macro for catching and re-throwing an exception of a given type.
- **#define AMQ_CATCH_EXCEPTION_CONVERT(sourceType, targetType)**
Macro for catching an exception of one type and then re-throwing as another type.
- **#define AMQ_CATCHALL_THROW(type)**
A catch-all that throws a known exception.
- **#define AMQ_CATCHALL_NOTHROW()**
A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

- `#define AMQ_CATCH_NOTHROW(type)`
Macro for catching and re-throwing an exception of a given type.

7.104.1 Define Documentation

7.104.1.1 `#define AMQ_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`

Value:

```
catch( sourceType& ex ){ \
    targetType target( ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then re-throwing as another type.

Parameters

sourceType the type of the exception to be caught.

targetType the type of the exception to be thrown.

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.104.1.2 `#define AMQ_CATCH_NOTHROW(type)`

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and re-throwing an exception of a given type.

Parameters

type The type of the exception to throw (e.g. `ActiveMQException`).

7.104.1.3 `#define AMQ_CATCH_RETHROW(type)`

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and re-throwing an exception of a given type.

Parameters

type The type of the exception to throw (e.g. `ActiveMQException`).

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.104.1.4 #define AMQ_CATCHALL_NOTHROW()**Value:**

```
catch( ... ){ \
    activemq::exceptions::ActiveMQException ex( __FILE__, __LINE__, \
        "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

7.104.1.5 #define AMQ_CATCHALL_THROW(type)**Value:**

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

Parameters

type the type of exception to be thrown.

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.105 src/main/decaf/lang/exceptions/ExceptionDefines.h

File Reference

Defines

- **#define DECAF_CATCH_RETHROW(type)**
Macro for catching and rethrowing an exception of a given type.
- **#define DECAF_CATCH_EXCEPTION_CONVERT(sourceType, targetType)**
Macro for catching an exception of one type and then rethrowing as another type.
- **#define DECAF_CATCHALL_THROW(type)**

A catch-all that throws a known exception.

- `#define DECAF_CATCHALL_NOTHROW()`
A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.
- `#define DECAF_CATCH_NOTHROW(type)`
Macro for catching and rethrowing an exception of a given type.

7.105.1 Define Documentation

7.105.1.1 `#define DECAF_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`

Value:

```
catch( sourceType& ex ){ \
    targetType target( &ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then rethrowing as another type.

Parameters

sourceType the type of the exception to be caught.

targetType the type of the exception to be thrown.

Referenced by `decaf::util::PriorityQueue< E >::add()`.

7.105.1.2 `#define DECAF_CATCH_NOTHROW(type)`

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and rethrowing an exception of a given type.

Parameters

type The type of the exception to throw (e.g. Exception).

7.105.1.3 `#define DECAF_CATCH_RETHROW(type)`

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and rethrowing an exception of a given type.

Parameters

type The type of the exception to throw (e.g. `Exception`).

Referenced by `decaf::util::PriorityQueue< E >::add()`.

7.105.1.4 `#define DECAF_CATCHALL_NOTHROW()`

Value:

```
catch( ... ){ \
    lang::Exception ex( __FILE__, __LINE__, \
        "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

7.105.1.5 `#define DECAF_CATCHALL_THROW(type)`

Value:

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

Parameters

type the type of exception to be thrown.

Referenced by `decaf::util::PriorityQueue< E >::add()`.

7.106 `src/main/activemq/io/LoggingInputStream.h` File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class `activemq::io::LoggingInputStream`

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::io**

7.107 src/main/activemq/io/LoggingOutputStream.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
```

Data Structures

- class **activemq::io::LoggingOutputStream**

OutputStream filter that just logs the data being written.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::io**

7.108 src/main/activemq/library/ActiveMQCPP.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::library::ActiveMQCPP**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::library**

7.109 src/main/activemq/state/CommandVisitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::state::CommandVisitor**

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**
- namespace **activemq::state**

7.110 src/main/activemq/state/CommandVisitorAdapter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/state/CommandVisitor.h>
#include <activemq/core/ActiveMQConstants.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/SessionId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/RemoveSubscriptionInfo.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageAck.h>
```

```
#include <activemq/commands/MessagePull.h>
#include <activemq/commands/TransactionInfo.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/commands/MessageDispatchNotification.h>
#include <activemq/commands/ControlCommand.h>
#include <activemq/commands/ConnectionError.h>
#include <activemq/commands/ConnectionControl.h>
#include <activemq/commands/ConsumerControl.h>
#include <activemq/commands/ShutdownInfo.h>
#include <activemq/commands/KeepAliveInfo.h>
#include <activemq/commands/FlushCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/commands/ReplayCommand.h>
#include <activemq/commands/Response.h>
```

Data Structures

- class **activemq::state::CommandVisitorAdapter**

*Default Implementation of a **CommandVisitor** (p. 1113) that returns NULL for all calls.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.111 src/main/activemq/state/ConnectionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/SessionInfo.h>
```

```
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::state::ConnectionState`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::state`

7.112 `src/main/activemq/state/ConnectionStateTracker.h` File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/state/CommandVisitorAdapter.h>
#include <activemq/state/ConnectionState.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
```

```
#include <activemq/state/Tracked.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::state::ConnectionStateTracker**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**

7.113 src/main/activemq/state/ConsumerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::state::ConsumerState**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**

7.114 src/main/activemq/state/ProducerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ProducerInfo.h>
```

```
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::state::ProducerState`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::state`

7.115 `src/main/activemq/state/SessionState.h` File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::state::SessionState`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::state`

7.116 src/main/activemq/state/Tracked.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class `activemq::state::Tracked`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::state`

7.117 src/main/activemq/state/TransactionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlList.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::state::TransactionState`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.118 src/main/activemq/threads/CompositeTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/Task.h>
```

Data Structures

- class **activemq::threads::CompositeTask**

Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p. 1133).

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.119 src/main/activemq/threads/CompositeTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTask.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::threads::CompositeTaskRunner**

A Task (p. 3494) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.120 src/main/activemq/threads/DedicatedTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/Task.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::threads::DedicatedTaskRunner**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.121 src/main/activemq/threads/Task.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::threads::Task**

Represents a unit of work that requires one or more iterations to complete.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.122 src/main/activemq/threads/TaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/Task.h>
```

Data Structures

- class **activemq::threads::TaskRunner**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.123 src/main/activemq/transport/AbstractTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportFactory.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::transport::AbstractTransportFactory**

*Abstract implementation of the **TransportFactory** (p. 3634) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3634) instances.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.124 src/main/activemq/transport/CompositeTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/List.h>
```

Data Structures

- class **activemq::transport::CompositeTransport**
*A **Composite Transport** (p. 3629) is a **Transport** (p. 3629) implementation that is composed of several **Transports**.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.125 src/main/activemq/transport/correlator/FutureResponse.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::transport::correlator::FutureResponse**

A container that holds a response object.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

7.126 src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/transport/correlator/FutureResponse.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <map>
#include <stdio.h>
```

Data Structures

- class **activemq::transport::correlator::ResponseCorrelator**

This type of transport filter is responsible for correlating asynchronous responses with requests.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

7.127 src/main/activemq/transport/DefaultTransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::DefaultTransportListener**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.128 src/main/activemq/transport/failover/BackupTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <memory>
```

Data Structures

- class **activemq::transport::failover::BackupTransport**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.129 src/main/activemq/transport/failover/BackupTransportPool.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/BackupTransport.h>
#include <activemq/transport/failover/URIPool.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/IOException.h>
#include <decaf/util/StlList.h>
```

Data Structures

- class **activemq::transport::failover::BackupTransportPool**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.130 src/main/activemq/transport/failover/CloseTransportsTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::failover::CloseTransportsTask**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.131 src/main/activemq/transport/failover/FailoverTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/state/ConnectionStateTracker.h>
#include <activemq/transport/CompositeTransport.h>
#include <activemq/transport/failover/BackupTransportPool.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/FailoverTransportListener.h>
#include <activemq/transport/failover/URIPool.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/StlList.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/net/URI.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransport**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.132 src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/transport/Transport.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransportFactory**
*Creates an instance of a **FailoverTransport** (p. 1752).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.133 src/main/activemq/transport/failover/FailoverTransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransportListener**
*Utility class used by the **Transport** (p. 3629) to perform the work of responding to events from the active **Transport** (p. 3629).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.134 src/main/activemq/transport/failover/URIPool.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/net/URI.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
```

Data Structures

- class **activemq::transport::failover::URIPool**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.135 src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Timer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
```

Data Structures

- class `activemq::transport::inactivity::InactivityMonitor`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::transport`
- namespace `activemq::transport::inactivity`

7.136 `src/main/activemq/transport/inactivity/ReadChecker.h` File Reference

```
#include <activemq/util/Config.h>
```

```
#include <decaf/util/TimerTask.h>
```

Data Structures

- class `activemq::transport::inactivity::ReadChecker`

Runnable class that is used by the {}.

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::transport`
- namespace `activemq::transport::inactivity`

7.137 `src/main/activemq/transport/inactivity/WriteChecker.h` File Reference

```
#include <activemq/util/Config.h>
```

```
#include <decaf/util/TimerTask.h>
```

Data Structures

- class `activemq::transport::inactivity::WriteChecker`

Runnable class used by the {}.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

7.138 src/main/activemq/transport/IOTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Thread.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <memory>
```

Data Structures

- class **activemq::transport::IOTransport**

*Implementation of the **Transport** (p. 3629) interface that performs marshaling of commands to IO streams.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.139 src/main/activemq/transport/logging/LoggingTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::logging::LoggingTransport**
A transport filter that logs commands as they are sent/received.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::logging**

7.140 src/main/activemq/transport/mock/InternalCommandListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
```

Data Structures

- class **activemq::transport::mock::InternalCommandListener**
*Listens for Commands sent from the **MockTransport** (p. 2592).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.141 src/main/activemq/transport/mock/MockTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/mock/InternalCommandListener.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <cms/Message.h>
#include <map>
#include <set>
```

Data Structures

- class **activemq::transport::mock::MockTransport**

*The **MockTransport** (p. 2592) defines a base level **Transport** (p. 3629) class that is intended to be used in place of an a regular protocol **Transport** (p. 3629) such as TCP.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.142 src/main/activemq/transport/mock/MockTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
```

Data Structures

- class **activemq::transport::mock::MockTransportFactory**
Manufactures MockTransports, which are objects that read from input streams and write to output streams.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.143 src/main/activemq/transport/mock/ResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
```

Data Structures

- class **activemq::transport::mock::ResponseBuilder**
Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.144 src/main/activemq/transport/tcp/SslTransport.h File Reference

```
#include <activemq/util/Config.h>  
#include <activemq/transport/tcp/TcpTransport.h>
```

Data Structures

- class **activemq::transport::tcp::SslTransport**
Transport (p. 3629) for connecting to a Broker using an SSL Socket.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.145 src/main/activemq/transport/tcp/SslTransportFactory.h File Reference

```
#include <activemq/util/Config.h>  
#include <activemq/transport/tcp/TcpTransportFactory.h>
```

Data Structures

- class **activemq::transport::tcp::SslTransportFactory**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.146 src/main/activemq/transport/tcp/TcpTransport.h File Reference

```
#include <activemq/io/LoggingInputStream.h>
#include <activemq/io/LoggingOutputStream.h>
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/net/Socket.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/BufferedInputStream.h>
#include <decaf/io/BufferedOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <memory>
```

Data Structures

- class **activemq::transport::tcp::TcpTransport**
*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2005).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.147 src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
```



```
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::transport::tcp::TcpTransportFactory**
*Factory Responsible for creating the **TcpTransport** (p. 3510).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.148 src/main/activemq/transport/Transport.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <typeinfo>
```

Data Structures

- class **activemq::transport::Transport**
Interface for a transport layer for command objects.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::transport**

7.149 src/main/activemq/transport/TransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::TransportFactory**
Defines the interface for Factories that create Transports or TransportFilters.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.150 src/main/activemq/transport/TransportFilter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
#include <typeinfo>
```

Data Structures

- class **activemq::transport::TransportFilter**
A filter on the transport layer.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.151 src/main/activemq/transport/TransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::TransportListener**

A listener of asynchronous exceptions from a command transport object.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.152 src/main/activemq/transport/TransportRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/transport/TransportFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::transport::TransportRegistry**

*Registry of all **Transport** (p. 3629) Factories that are available to the client at runtime.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.153 src/main/activemq/util/ActiveMQProperties.h File Reference

```
#include <map>
#include <string>
#include <sstream>
#include <activemq/util/Config.h>
#include <cms/CMSProperties.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::util::ActiveMQProperties**

*Implementation of the **CMSProperties** interface that delegates to a **decaf::util::Properties** (p. 2927) object.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.154 src/main/activemq/util/CMSExceptionSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
```

```

#include <cms/CMSSecurityException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/InvalidClientIdException.h>
#include <cms/InvalidDestinationException.h>
#include <cms/InvalidSelectorException.h>
#include <cms/IllegalStateException.h>
#include <cms/UnsupportedOperationException.h>
#include <decaf/lang/Exception.h>
#include <string>

```

Data Structures

- class `activemq::util::CMSExceptionSupport`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::util`

Defines

- `#define AMQ_CATCH_ALL_THROW_CMSEXCEPTION()`

Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.

7.154.1 Define Documentation

7.154.1.1 `#define AMQ_CATCH_ALL_THROW_CMSEXCEPTION()`

Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.

Referenced by	<code>activemq::commands::ActiveMQMessageTemplate<</code>	<code>cms::ObjectMessage</code>
<code>>::acknowledge(),</code>	<code>activemq::commands::ActiveMQMessageTemplate<</code>	<code>cms::ObjectMessage</code>
<code>>::clearBody(),</code>	<code>activemq::commands::ActiveMQMessageTemplate<</code>	<code>cms::ObjectMessage</code>
<code>>::clearProperties(),</code>	<code>activemq::commands::ActiveMQMessageTemplate<</code>	<code>cms::ObjectMessage</code>
<code>>::equals(),</code>	<code>activemq::commands::ActiveMQMessageTemplate<</code>	<code>cms::ObjectMessage</code>

```

>::getBooleanProperty(),      activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getBytesProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getCMSMessageID(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getDoubleProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getFloatProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getIntProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getLongProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getPropertyNames(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getShortProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getStringProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::propertyExists(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setBooleanProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setBytesProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setCMSDestination(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setCMSReplyTo(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setDoubleProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setFloatProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setIntProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setLongProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setShortProperty(), and activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setStringProperty().

```

7.155 src/main/activemq/util/CompositeData.h File Reference

```

#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/net/URI.h>
#include <decaf/net/URISyntaxException.h>

```

Data Structures

- class **activemq::util::CompositeData**

Represents a Composite URI.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.156 src/main/activemq/util/Config.h File Reference

Defines

- `#define AMQCPP_API`
- `#define HAVE_UUID_UUID_H`
- `#define HAVE_UUID_T`
- `#define HAVE_PTHREAD_H`

7.156.1 Define Documentation

7.156.1.1 `#define AMQCPP_API`

7.156.1.2 `#define HAVE_PTHREAD_H`

7.156.1.3 `#define HAVE_UUID_T`

7.156.1.4 `#define HAVE_UUID_UUID_H`

7.157 src/main/cms/Config.h File Reference

Defines

- `#define CMS_API`

7.157.1 Define Documentation

7.157.1.1 `#define CMS_API`

7.158 src/main/decaf/util/Config.h File Reference

Defines

- `#define DECAF_API`
- `#define HAVE_UUID_UUID_H`
- `#define HAVE_UUID_T`
- `#define HAVE_PTHREAD_H`
- `#define DECAF_UNUSED`

7.158.1 Define Documentation

7.158.1.1 `#define DECAF_API`

7.158.1.2 `#define DECAF_UNUSED`

7.158.1.3 `#define HAVE_PTHREAD_H`

7.158.1.4 `#define HAVE_UUID_T`

7.158.1.5 `#define HAVE_UUID_UUID_H`

7.159 `src/main/activemq/util/IdGenerator.h` File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
#include <string>
```

Data Structures

- class `activemq::util::IdGenerator`
- class `activemq::util::IdGenerator::StaticData`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::util`

7.160 `src/main/activemq/util/LongSequenceGenerator.h` File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class `activemq::util::LongSequenceGenerator`

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.161 src/main/activemq/util/MarshallingSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <string>
```

Data Structures

- class **activemq::util::MarshallingSupport**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.162 src/main/activemq/util/MemoryUsage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::util::MemoryUsage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.163 src/main/activemq/util/PrimitiveList.h File Reference

```
#include <string>
#include <vector>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <stdio.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

Data Structures

- class **activemq::util::PrimitiveList**

List of primitives.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.164 src/main/activemq/util/PrimitiveMap.h File Reference

```
#include <string>
#include <vector>
#include <activemq/util/Config.h>
#include <decaf/util/Config.h>
#include <decaf/util/StlMap.h>
```

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

Data Structures

- class **activemq::util::PrimitiveMap**
Map of named primitives.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.165 src/main/activemq/util/PrimitiveValueConverter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <string>
```

Data Structures

- class **activemq::util::PrimitiveValueConverter**
*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2817) from one type to another.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.166 src/main/activemq/util/PrimitiveValueNode.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Map.h>
#include <decaf/util/List.h>
```

Data Structures

- class **activemq::util::PrimitiveValueNode**
Class that wraps around a single value of one of the many types.
- union **activemq::util::PrimitiveValueNode::PrimitiveValue**
Define a union type comprised of the various types.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.167 src/main/activemq/util/URISupport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/CompositeData.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::util::URISupport**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.168 src/main/activemq/util/Usage.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class `activemq::util::Usage`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::util`

7.169 src/main/activemq/wireformat/MarshalAware.h File Reference

```
#include <vector>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
```

Data Structures

- class `activemq::wireformat::MarshalAware`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`

7.170 src/main/activemq/wireformat/openwire/marshal/BaseDataStream File Reference

```
#include <activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
#include <activemq/wireformat/openwire/utills/HexTable.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller**
Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.171 src/main/activemq/wireformat/openwire/marshal/DataStreamM File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
Base class for all classes that marshal commands for Openwire.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.172

src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File

Reference

7.172 src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/util/PrimitiveList.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/IOException.h>
#include <string>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller**

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.173 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQWireFormat.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 174).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v1`

7.174 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 182).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v2`

7.175 src/main/activemq/wireformat/openwire/marsh
shal/v3/ActiveMQBlobMessageMarshaller.h File

Reference

7.175 src/main/activemq/wireformat/openwire/marsh³⁹⁰⁷/v3/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marsh::v3::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 171).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marsh**
- namespace **activemq::wireformat::openwire::marsh::v3**

7.176 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 178).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v4`

7.177 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 186).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v5`

7.178 src/main/activemq/wireformat/openwire/marsh-
shal/v6/ActiveMQBlobMessageMarshaller.h File

Reference

7.178 src/main/activemq/wireformat/openwire/marsh³⁹⁰⁹/v6/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshall/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 190).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.179 src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshall/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 213).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.180 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 229).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.181 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File

Reference

7.181 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 210).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marsh**
- namespace **activemq::wireformat::openwire::marsh::v3**

7.182 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 217).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v4`

7.183 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 221).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v5`

7.184 src/main/activemq/wireformat/openwire/marsh- shal/v6/ActiveMQBytesMessageMarshaller.h File

Reference

7.184 src/main/activemq/wireformat/openwire/marsh³⁹¹³/v6/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 225).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.185 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 293).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v1`

7.186 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 305).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v2`

7.187 src/main/activemq/wireformat/openwire/marsh-
shal/v3/ActiveMQDestinationMarshaller.h File

Reference

7.187 src/main/activemq/wireformat/openwire/marsh³⁹¹⁵/v3/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshall::v3::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 289).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshall**
- namespace **activemq::wireformat::openwire::marshall::v3**

7.188 src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 297).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v4`

7.189 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 301).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v5`

7.190 src/main/activemq/wireformat/openwire/marsh-
shal/v6/ActiveMQDestinationMarshaller.h File

Reference

7.190 src/main/activemq/wireformat/openwire/marsh³⁹¹⁷/v6/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 309).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.191 src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshall/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 332).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v1`

7.192 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 344).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v2`

7.193 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h File

Reference

7.193 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 328).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marsh**
- namespace **activemq::wireformat::openwire::marsh::v3**

7.194 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 336).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v4`

7.195 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 340).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v5`

7.196 `src/main/activemq/wireformat/openwire/mar-`
`shal/v6/ActiveMQMapMessageMarshaller.h` File

Reference

7.196 `src/main/activemq/wireformat/openwire/marsh`³⁹²¹`al/v6/ActiveMQ`
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 348).

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v6`

7.197 `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQ`
File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 359).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.198 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 371).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.199 src/main/activemq/wireformat/openwire/marsh
shal/v3/ActiveMQMessageMarshaller.h File

Reference

7.199 src/main/activemq/wireformat/openwire/marsh³⁹²³ shal/v3/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 355).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.200 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 363).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.201 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 367).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.202 src/main/activemq/wireformat/openwire/marsh
shal/v6/ActiveMQMessageMarshaller.h File

Reference

7.202 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h File Reference ³⁹²⁵

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 375).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.203 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 403).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v1`

7.204 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 415).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v2`

7.205 src/main/activemq/wireformat/openwire/marsh
shal/v3/ActiveMQObjectMessageMarshaller.h File

Reference

7.205 src/main/activemq/wireformat/openwire/marsh³⁹²⁷ shal/v3/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 399).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.206 src/main/activemq/wireformat/openwire/marsh shal/v4/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 407).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v4`

7.207 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 411).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v5`

7.208 src/main/activemq/wireformat/openwire/marsh
shal/v6/ActiveMQObjectMessageMarshaller.h File

Reference

7.208 ³⁹²⁹src/main/activemq/wireformat/openwire/marsh/v6/ActiveMQO
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marsh::v6::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 419).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marsh**
- namespace **activemq::wireformat::openwire::marsh::v6**

7.209 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQO
File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller**
Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 445).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.210 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller**
Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 457).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.211

src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h

File Reference

7.211 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 441).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.212 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller**
Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 449).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.213 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller**
Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 453).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.214

src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h

File Reference

7.214 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h 3933

File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 461).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.215 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 503).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v1`

7.216 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 515).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v2`

7.217 src/main/activemq/wireformat/openwire/marsh- shal/v3/ActiveMQStreamMessageMarshaller.h File

Reference

7.217 src/main/activemq/wireformat/openwire/marsh³⁹³⁵/v3/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshall/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshall::v3::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 499).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshall**
- namespace **activemq::wireformat::openwire::marshall::v3**

7.218 src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshall/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 507).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v4`

7.219 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 511).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v5`

7.220 src/main/activemq/wireformat/openwire/marsh- shal/v6/ActiveMQStreamMessageMarshaller.h File

Reference

7.220 src/main/activemq/wireformat/openwire/marsh³⁹³⁷/v6/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 519).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.221 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQ7 File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 530).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v1`

7.222 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 541).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`

7.223 src/main/activemq/wireformat/openwire/marsh shal/v3/ActiveMQTempDestinationMarshaller.h File Reference

3939

- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.223 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller**
(p. 527).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.224 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/DataSet.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 534).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.225 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/Util/Config.h>
#include <activemq/commands/DataSet.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 538).*

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.226 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>  
#include <decaf/io/DataInputStream.h>  
#include <decaf/io/DataOutputStream.h>  
#include <decaf/io/IOException.h>  
#include <activemq/util/Config.h>  
#include <activemq/commands/DataStructure.h>  
#include <activemq/wireformat/openwire/OpenWireFormat.h>  
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller**
(p. 545).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.227 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 556).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.228 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 568).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.229 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 552).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.230 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 560).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.231 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 564).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.232 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 572).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.233 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 588).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.234 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 596).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.235 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 580).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.236 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 584).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.237 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 592).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.238 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 600).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.239 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 617).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.240 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 629).*

Namespaces

- namespace `activemq`
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v2`

7.241 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 609).*

Namespaces

- namespace `activemq`
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v3`

7.242 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 613).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.243 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 621).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.244 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 625).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.245 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 644).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.246 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 656).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.247 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 636).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.248 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 640).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.249 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 648).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.250 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 652).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.251 src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 714).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.252 src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.253

src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h

File Reference

3959

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 734).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.253 src/main/activemq/wireformat/openwire/marshal/v3/BaseComm

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 701).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.254 src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 708).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.255 src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 721).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.256 src/main/activemq/wireformat/openwire/marshal/v6/BaseComm

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 728).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.257 src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 808).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.258 src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 820).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.259 src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 801).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.260 src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 805).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.261 src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 812).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.262 src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 816).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.263 src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 839).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.264 src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 851).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.265 src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 831).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.266 src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 835).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.267 src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 843).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.268 src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 847).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.269 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1184).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.270 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1196).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.271 src/main/activemq/wireformat/openwire/marshal/v3/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1177).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.272 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1180).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.273 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1188).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.274 src/main/activemq/wireformat/openwire/marshal/v6/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1192).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.275 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1216).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.276 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1204).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.277 src/main/activemq/wireformat/openwire/marshal/v3/Connection

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1208).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.278 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1212).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.279 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1220).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.280 src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1224).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.281 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1245).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.282 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1234).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.283 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utills/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1238).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.284 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1241).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.285 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1249).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.286 src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1253).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.287 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1275).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.288 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1263).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.289 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1267).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.290 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1271).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.291 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1279).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.292 src/main/activemq/wireformat/openwire/marshal/v6/ConnectionInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1283).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.293 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1318).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.294 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1306).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.295 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerC File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utis/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1310).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.296 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1314).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.297 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1322).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.298 src/main/activemq/wireformat/openwire/marshal/v6/ConsumerC File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1326).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.299 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1345).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.300 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.301

src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h File Reference

3991

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1334).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.301 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1338).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.302 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1342).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.303 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.304

src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h File Reference

3993

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1349).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.304 src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1353).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.305 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1377).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.306 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


7.307

src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h

File Reference

3995

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1366).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.307 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1369).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.308 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1373).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.309 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.310

src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h

File Reference

3997

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1381).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.310 src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1385).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.311 src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1405).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.312 src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1393).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.313 src/main/activemq/wireformat/openwire/marshal/v3/ControlCo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>  
#include <decaf/io/DataInputStream.h>  
#include <decaf/io/DataOutputStream.h>  
#include <decaf/io/IOException.h>  
#include <activemq/util/Config.h>  
#include <activemq/commands/DataStructure.h>  
#include <activemq/wireformat/openwire/OpenWireFormat.h>  
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1397).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.314 src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1401).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.315 src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1409).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.316 src/main/activemq/wireformat/openwire/marshal/v6/ControlCo File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1413).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.317 src/main/activemq/wireformat/openwire/marshal/v1/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1438).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.318 src/main/activemq/wireformat/openwire/marshal/v2/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1426).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.319 src/main/activemq/wireformat/openwire/marshal/v3/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1430).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.320 src/main/activemq/wireformat/openwire/marshal/v4/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1434).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.321 src/main/activemq/wireformat/openwire/marshal/v5/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1442).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.322 src/main/activemq/wireformat/openwire/marshal/v6/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1446).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.323 src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1499).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.324 src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1487).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.325 src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1491).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.326 src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1495).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.327 src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.328

src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h

File Reference

4009

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1479).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.328 src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1483).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.329 src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1630).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.330 src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1618).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.331 src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1622).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.332 src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1626).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.333 src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.334

src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h

File Reference

4013

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1638).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.334 src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1634).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.335 src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1662).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.336 src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1651).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.337 src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utis/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1654).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.338 src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1658).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.339 src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.340

src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller.h

File Reference

4017

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1666).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.340 src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1647).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.341 src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1742).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.342 src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1726).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.343 src/main/activemq/wireformat/openwire/marshal/v3/ExceptionR File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1730).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.344 src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1738).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.345 src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1734).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.346 src/main/activemq/wireformat/openwire/marshal/v6/ExceptionR File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1722).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.347 src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1830).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.348 src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1818).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.349 src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1822).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.350 src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1826).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.351 src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1834).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.352 src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1814).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.353 src/main/activemq/wireformat/openwire/marshal/v1/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1974).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.354 src/main/activemq/wireformat/openwire/marshal/v2/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1962).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.355 src/main/activemq/wireformat/openwire/marshal/v3/IntegerRes

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1966).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.356 src/main/activemq/wireformat/openwire/marshal/v4/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1970).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.357 src/main/activemq/wireformat/openwire/marshal/v5/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1978).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.358 src/main/activemq/wireformat/openwire/marshal/v6/IntegerRes

File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1958).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.359 src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2036).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.360 src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2021).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.361 src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2028).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.362 src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2032).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.363 src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2025).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.364 src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2017).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.365 src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2065).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.366 src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2049).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.367 src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2053).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.368 src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2061).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.369 src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2045).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.370 src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2057).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.371 src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2087).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.372 src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2072).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.373 src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2076).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.374 src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2083).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.375 src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.376

src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h

File Reference

4041

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2091).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.376 src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utls/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2079).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.377 src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2118).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.378 src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2102).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.379 src/main/activemq/wireformat/openwire/marshal/v3/JournalTra File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2106).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.380 src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2114).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.381 src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2110).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.382 src/main/activemq/wireformat/openwire/marshal/v6/JournalTra File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2099).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.383 src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2145).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.384 src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2129).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.385 src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2133).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.386 src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2137).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.387 src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2141).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.388 src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2125).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.389 src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2178).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.390 src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2166).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.391 src/main/activemq/wireformat/openwire/marshal/v3/LastPartial File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2162).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.392 src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2174).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.393 src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2170).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.394 src/main/activemq/wireformat/openwire/marshal/v6/LastPartial File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2158).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.395 src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2224).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.396 src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2208).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.397 src/main/activemq/wireformat/openwire/marshal/v3/LocalTrans File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2212).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.398 src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2220).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.399 src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2216).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.400 src/main/activemq/wireformat/openwire/marshal/v6/LocalTrans File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2204).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.401 src/main/activemq/wireformat/openwire/marshal/v1/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.402 src/main/activemq/wireformat/openwire/marshal/v2/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.403 src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.404 src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.405 src/main/activemq/wireformat/openwire/marshal/v5/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.406 src/main/activemq/wireformat/openwire/marshal/v6/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.407

src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h File

Reference

7.407 src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h 4061

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2415).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.408 src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2403).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.409 src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2407).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.410

src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h File

Reference

7.410 src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h 4063

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2411).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.411 src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2419).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.412 src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2399).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.413 src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h File

Reference

7.413 src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2454).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marsh**
- namespace **activemq::wireformat::openwire::marsh::v1**

7.414 src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2438).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.415 src/main/activemq/wireformat/openwire/marshal/v3/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2442).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.416 src/main/activemq/wireformat/openwire/marsh shal/v4/MessageDispatchMarshaller.h File

Reference

7.416 src/main/activemq/wireformat/openwire/marsh⁴⁰⁶⁷shal/v4/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marsh::v4::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2450).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marsh**
- namespace **activemq::wireformat::openwire::marsh::v4**

7.417 src/main/activemq/wireformat/openwire/marshal/v5/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2446).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.418 src/main/activemq/wireformat/openwire/marshal/v6/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2458).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.419 src/main/activemq/wireformat/openwire/marsh
shal/v1/MessageDispatchNotificationMarshaller.h File

Reference

7.419 src/main/activemq/wireformat/openwire/marsh⁴⁰⁶⁹shal/v1/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller**
(p. 2482).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.420 src/main/activemq/wireformat/openwire/marshal/v2/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`

Marshaling code for Open Wire Format for `MessageDispatchNotificationMarshaller` (p. 2470).

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v2`

7.421 `src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h` File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`

Marshaling code for Open Wire Format for `MessageDispatchNotificationMarshaller` (p. 2474).

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`

7.422 src/main/activemq/wireformat/openwire/marsh shal/v4/MessageDispatchNotificationMarshaller.h File

Reference

4071

- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.422 src/main/activemq/wireformat/openwire/marshal/v4/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller**
(p. 2478).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.423 src/main/activemq/wireformat/openwire/marshal/v5/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/DataSet.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**

Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2486).

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.424 src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/Util/Config.h>
#include <activemq/commands/DataSet.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**

Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2466).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.425 src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2518).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.426 src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2499).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.427 src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2510).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.428 src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2503).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.429 src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2507).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.430 src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2514).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.431 src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2540).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.432 src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageMarshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2532).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.433 src/main/activemq/wireformat/openwire/marshal/v3/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2527).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.434 src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2536).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.435 src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageMarshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2523).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.436 src/main/activemq/wireformat/openwire/marshal/v6/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageMarshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2544).*

7.437

src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h

File Reference

4081

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.437 src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2584).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.438 src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2568).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.439 src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2576).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.440 src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2580).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.441 src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2572).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.442 src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2588).*

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.443 src/main/activemq/wireformat/openwire/marshal/v1/NetworkBr File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2636).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.444 src/main/activemq/wireformat/openwire/marshal/v2/NetworkBr File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2617).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.445 src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2628).*

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.446 src/main/activemq/wireformat/openwire/marshal/v4/NetworkBr File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2632).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.447 src/main/activemq/wireformat/openwire/marshal/v5/NetworkBr File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2624).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.448 src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2620).*

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.449 src/main/activemq/wireformat/openwire/marshal/v1/PartialCon File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2752).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.450 src/main/activemq/wireformat/openwire/marshal/v2/PartialCon File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2735).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.451 src/main/activemq/wireformat/openwire/marshal/v3/PartialCon File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2744).*

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.452 src/main/activemq/wireformat/openwire/marshal/v4/PartialCon File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2748).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.453 src/main/activemq/wireformat/openwire/marshal/v5/PartialCon File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2739).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.454 src/main/activemq/wireformat/openwire/marshal/v6/PartialCon File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2731).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.455 src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2862).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.456 src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2842).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.457 src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2850).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.458 src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2846).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.459 src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2854).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.460 src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2858).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.461 src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2893).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.462 src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2874).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.463 src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2881).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.464 src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2878).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.465 src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2885).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.466 src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2889).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.467 src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2910).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.468 src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2906).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.469 src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2918).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.470 src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2902).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.471 src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2914).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.472 src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2922).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.473 src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3003).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.474 src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2991).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.475 src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2999).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.476 src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3011).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.477 src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3007).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.478 src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2995).*

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.479 src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3019).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.480 src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3027).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.481 src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3023).*

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.482 src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3039).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.483 src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3035).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.484 src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3031).*

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.485 src/main/activemq/wireformat/openwire/marshal/v1/ReplayCon File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller**
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3050).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.486 src/main/activemq/wireformat/openwire/marshal/v2/ReplayCon File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3054).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.487 src/main/activemq/wireformat/openwire/marshal/v3/ReplayCon File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3058).*

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.488 src/main/activemq/wireformat/openwire/marshal/v4/ReplayCon File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3046).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.489 src/main/activemq/wireformat/openwire/marshal/v5/ReplayCon File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3066).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.490 src/main/activemq/wireformat/openwire/marshal/v6/ReplayCon File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3062).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.491 src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3102).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.492 src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3089).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.493 src/main/activemq/wireformat/openwire/marshal/v3/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3098).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.494 src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3085).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.495 src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3093).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.496 src/main/activemq/wireformat/openwire/marshal/v6/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3107).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.497 src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3184).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.498 src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3165).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.499 src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3180).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.500 src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3169).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.501 src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3176).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.502 src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3173).*

7.503

src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h **File Reference** **4125**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.503 src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3199).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.504 src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3207).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.505 src/main/activemq/wireformat/openwire/marshal/v3/SessionInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3203).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.506 src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3211).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.507 src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3195).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.508 src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoFile Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3192).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.509 src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller**

Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 3260).

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.510 src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3256).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.511 src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3268).*

7.512

src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h

File Reference

4131

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.512 src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller**

Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 3272).

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.513 src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3264).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.514 src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3252).*

7.515

src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h

File Reference

4133

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.515 src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3442).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.516 src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3457).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.517 src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3438).*

7.518

src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h

File Reference

4135

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.518 src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3450).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.519 src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3446).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.520 src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3453).*

7.521

src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h

File Reference

4137

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.521 src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3576).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.522 src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3580).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.523 src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3583).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.524 src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3587).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.525 src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3572).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.526 src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3591).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.527 src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3602).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.528 src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3618).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.529 src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3606).*

7.530

`src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h`

File Reference

4143

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.530 `src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h` File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3614).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.531 `src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h` File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3598).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.532 src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3610).*

7.533

src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h

File Reference

4145

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.533 src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3743).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.534 src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3735).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.535 src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3747).*

7.536

src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h

File Reference

4147

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.536 src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3739).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.537 src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3728).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.538 src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3731).*

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.539 src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller**
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3781).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.540 src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3773).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.541 src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3785).*

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.542 src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller**
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3777).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.543 src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3789).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.544 src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3769).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.545 src/main/activemq/wireformat/openwire/OpenWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <memory>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormat**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.546 src/main/activemq/wireformat/openwire/OpenWireFormatFactor File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/commands/WireFormatInfo.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatFactory**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.547 src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatNegotiator**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.548 src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireResponseBuilder**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.549 src/main/activemq/wireformat/openwire/utils/BooleanStream.h File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::utils::BooleanStream**
Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.550 src/main/activemq/wireformat/openwire/Utils/HexTable.h File Reference

```
#include <vector>
#include <string>
#include <activemq/Util/Config.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **activemq::wireformat::openwire::Utils::HexTable**

*The **HexTable** (p. 1857) class maps hexadecimal strings to the value of an index into the table, i.e.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::Utils**

7.551 src/main/activemq/wireformat/openwire/Utils/MessagePropertyInterceptor.h File Reference

```
#include <activemq/Util/Config.h>
#include <activemq/Commands/Message.h>
#include <activemq/Util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **activemq::wireformat::openwire::Utils::MessagePropertyInterceptor**

*Used the base **ActiveMQMessage** class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the **OpenWireMessage** properties.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.552 src/main/activemq/wireformat/stomp/StompCommandConstants.h File Reference

```
#include <cms/Destination.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <string>
#include <map>
```

Data Structures

- class **activemq::wireformat::stomp::StompCommandConstants**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.553 src/main/activemq/wireformat/stomp/StompFrame.h File Reference

```
#include <string>
#include <string.h>
#include <map>
#include <decaf/util/Properties.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompFrame**

A Stomp-level message frame that encloses all messages to and from the broker.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.554 src/main/activemq/wireformat/stomp/StompHelper.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompHelper**

Utility Methods used when marshaling to and from StompFrame's.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.555 src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/wireformat/stomp/StompHelper.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompWireFormat**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.556 src/main/activemq/wireformat/stomp/StompWireFormatFactory File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/wireformat/stomp/StompWireFormat.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompWireFormatFactory**
Factory used to create the Stomp Wire Format instance.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.557 src/main/activemq/wireformat/WireFormat.h File Reference

```
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/Transport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **activemq::wireformat::WireFormat**
Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

7.558 src/main/activemq/wireformat/WireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **activemq::wireformat::WireFormatFactory**

*The **WireFormatFactory** (p. 3716) is the interface that all **WireFormatFactory** (p. 3716) classes must extend.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.559 src/main/activemq/wireformat/WireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::WireFormatNegotiator**

*Defines a **WireFormatNegotiator** (p. 3751) which allows a **WireFormat** (p. 3712) to.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.560 src/main/activemq/wireformat/WireFormatRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/wireformat/WireFormatFactory.h>
#include <decaf/util/StlMap.h>
```

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::wireformat::WireFormatRegistry**

*Registry of all **WireFormat** (p. 3712) Factories that are available to the client at runtime.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.561 src/main/cms/BytesMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWritableException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
```

Data Structures

- class **cms::BytesMessage**

*A **BytesMessage** (p. 979) object is used to send a message containing a stream of unsigned bytes.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.562 src/main/cms/Closeable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Closeable**
Interface for a class that implements the close method.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.563 src/main/decaf/io/Closeable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::Closeable**
Interface for a class that implements the close method.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.564 src/main/cms/CMSException.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <cms/Config.h>
```

Data Structures

- class **cms::CMSException**

CMS API Exception that is the base for all exceptions thrown from CMS classes.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.565 src/main/cms/CMSProperties.h File Reference

```
#include <cms/Config.h>
#include <map>
#include <string>
#include <vector>
```

Data Structures

- class **cms::CMSProperties**

Interface for a Java-like properties object.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.566 src/main/cms/CMSSecurityException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::CMSSecurityException**

This exception must be thrown when a provider rejects a user name/password submitted by a client.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.567 src/main/cms/Connection.h File Reference

```
#include <cms/Config.h>
#include <cms/Startable.h>
#include <cms/Stopable.h>
#include <cms/Closeable.h>
#include <cms/Session.h>
#include <cms/ConnectionMetaData.h>
```

Data Structures

- class **cms::Connection**

The client's connection to its provider.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.568 src/main/cms/ConnectionFactory.h File Reference

```
#include <cms/Config.h>
#include <cms/Connection.h>
#include <cms/CMSException.h>
#include <string>
```

Data Structures

- class **cms::ConnectionFactory**

*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1168) objects returned implement the **CMS Connection** (p. 1168) interface and hide the CMS Provider specific implementation details behind that interface.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.569 src/main/cms/ConnectionMetaData.h File Reference

```
#include <cms/Config.h>
```

```
#include <cms/CMSException.h>
```

Data Structures

- class **cms::ConnectionMetaData**

*A **ConnectionMetaData** (p. 1287) object provides information describing the **Connection** (p. 1168) object.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.570 src/main/cms/DeliveryMode.h File Reference

```
#include <cms/Config.h>
```

Data Structures

- class **cms::DeliveryMode**

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.571 src/main/cms/Destination.h File Reference

```
#include <cms/CMSProperties.h>
#include <cms/Config.h>
#include <string>
```

Data Structures

- class **cms::Destination**

*A **Destination** (p. 1610) object encapsulates a provider-specific address.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.572 src/main/cms/ExceptionListener.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::ExceptionListener**

*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1719) that is registered with the **Connection** (p. 1168).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.573 src/main/cms/IllegalStateException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::IllegalStateException**

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.574 src/main/decaf/lang/exceptions/IllegalStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalStateException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.575 src/main/cms/InvalidClientIdException.h File Reference

```
#include <cms/Config.h>  
#include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidClientIdException**

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.576 src/main/cms/InvalidDestinationException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidDestinationException**

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.577 src/main/cms/InvalidSelectorException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidSelectorException**

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.578 src/main/cms/MapMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class **cms::MapMessage**
*A **MapMessage** (p. 2318) object is used to send a set of name-value pairs.*

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.579 src/main/cms/MessageConsumer.h File Reference

```
#include <cms/Config.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/Closeable.h>
```

Data Structures

- class **cms::MessageConsumer**
*A client uses a **MessageConsumer** (p. 2423) to received messages from a destination.*

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.580 src/main/cms/MessageEnumeration.h File Reference

```
#include <cms/Config.h>
```



```
#include <cms/Message.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageEnumeration**

Defines an object that enumerates a collection of Messages.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.581 src/main/cms/MessageEOFException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3415) or **BytesMessage** (p. 979) is being read.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.582 src/main/cms/MessageFormatException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class `cms::MessageFormatException`

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

Namespaces

- namespace `cms`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.583 `src/main/cms/MessageListener.h` File Reference

```
#include <cms/Config.h>
```

Data Structures

- class `cms::MessageListener`

A `MessageListener` (p. 2522) object is used to receive asynchronously delivered messages.

Namespaces

- namespace `cms`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.584 `src/main/cms/MessageNotReadableException.h` File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class `cms::MessageNotReadableException`

This exception must be thrown when a CMS client attempts to read a write-only message.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.585 src/main/cms/MessageNotWriteableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageNotWriteableException**

This exception must be thrown when a CMS client attempts to write to a read-only message.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.586 src/main/cms/MessageProducer.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/Closeable.h>
#include <cms/CMSException.h>
#include <cms/InvalidDestinationException.h>
#include <cms/MessageFormatException.h>
#include <cms/UnsupportedOperationException.h>
#include <cms/DeliveryMode.h>
```

Data Structures

- class **cms::MessageProducer**

*A client uses a **MessageProducer** (p. 2550) object to send messages to a **Destination** (p. 1610).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.587 src/main/cms/ObjectMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
```

Data Structures

- class **cms::ObjectMessage**

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.588 src/main/cms/Queue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Queue**

An interface encapsulating a provider-specific queue name.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.589 src/main/decaf/util/Queue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::util::Queue< E >**
A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.590 src/main/cms/QueueBrowser.h File Reference

```
#include <string>
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Queue.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageEnumeration.h>
```

Data Structures

- class **cms::QueueBrowser**
*This class implements in interface for browsing the messages in a **Queue** (p. 2947) without removing them.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.591 src/main/cms/Session.h File Reference

```
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Message.h>
#include <cms/TextMessage.h>
#include <cms/BytesMessage.h>
#include <cms/MapMessage.h>
#include <cms/StreamMessage.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <cms/Topic.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/TemporaryTopic.h>
#include <cms/TemporaryQueue.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Session**

*A **Session** (p. 3148) object is a single-threaded context for producing and consuming messages.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.592 src/main/cms/Startable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Startable**

Interface for a class that implements the start method.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.593 src/main/cms/Stopable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Stoppable**

Interface for a class that implements the stop method.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.594 src/main/cms/StreamMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWritableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
```

Data Structures

- class **cms::StreamMessage**

*Interface for a **StreamMessage** (p. 3415).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.595 src/main/cms/TemporaryQueue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TemporaryQueue**

*Defines a Temporary **Queue** (p. 2947) based **Destination** (p. 1610).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.596 src/main/cms/TemporaryTopic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TemporaryTopic**

*Defines a Temporary **Topic** (p. 3568) based **Destination** (p. 1610).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.597 src/main/cms/TextMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class **cms::TextMessage**
Interface for a text message.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.598 src/main/cms/Topic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Topic**
An interface encapsulating a provider-specific topic name.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.599 src/main/cms/UnsupportedOperationException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::UnsupportedOperationException**

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.600 src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::UnsupportedOperationException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.601 src/main/decaf/internal/AprPool.h File Reference

```
#include <decaf/util/Config.h>
#include <apr_pools.h>
```

Data Structures

- class **decaf::internal::AprPool**

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**

7.602 src/main/decaf/internal/DecafRuntime.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runtime.h>
#include <apr_pools.h>
```

Data Structures

- class **decaf::internal::DecafRuntime**

Handles APR initialization and termination.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**

7.603 src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

Data Structures

- class **decaf::internal::io::StandardErrorOutputStream**

Wrapper Around the Standard error Output facility on the current platform.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.604 src/main/decaf/internal/io/StandardInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
```

Data Structures

- class **decaf::internal::io::StandardInputStream**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.605 src/main/decaf/internal/io/StandardOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

Data Structures

- class **decaf::internal::io::StandardOutputStream**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.606 src/main/decaf/internal/net/DefaultServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::DefaultServerSocketFactory**

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.607 src/main/decaf/internal/net/DefaultSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketFactory.h>
```

Data Structures

- class **decaf::internal::net::DefaultSocketFactory**

SocketFactory implementation that is used to create Sockets.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.608 src/main/decaf/internal/net/Network.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/internal/util/Resource.h>
#include <decaf/internal/util/GenericResource.h>
```

Data Structures

- class **decaf::internal::net::Network**

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.609 src/main/decaf/internal/net/SocketFileDescriptor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FileDescriptor.h>
```

Data Structures

- class **decaf::internal::net::SocketFileDescriptor**

File Descriptor type used internally by Decaf Socket objects.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.610 src/main/decaf/internal/net/ssl/DefaultSSLContext.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContext.h>
```

Data Structures

- class **decaf::internal::net::ssl::DefaultSSLContext**

Default SSL Context manager for the Decaf library.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

7.611 src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::ssl::DefaultSSLServerSocketFactory**

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

7.612 src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocketFactory.h>
#include <string>
#include <vector>
```

Data Structures

- class **decaf::internal::net::ssl::DefaultSSLSocketFactory**

Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

7.613 src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContextSpi.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLContextSpi**

Provides an SSLContext that wraps the OpenSSL API.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.614 src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <string>
```

```
#include <vector>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLParameters**

Container class for parameters that are Common to OpenSSL socket classes.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.615 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/net/ssl/SSLServerSocket.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocket**

SSLServerSocket based on OpenSSL library code.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.616 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/net/ssl/SSLServerSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory**

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.617 src/main/decaf/internal/net/ssl/openssl/SSLSocket.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/net/ssl/SSLSocket.h>
```

```
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocket**

Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.618 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketException**

Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.619 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory**
Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.620 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream**
An output stream for reading data from an OpenSSL Socket instance.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.621 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream**
*OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2673) instance.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.622 src/main/decaf/internal/net/tcp/TcpSocket.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/net/SocketImpl.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
#include <apr_network_io.h>
#include <decaf/io/IOException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::internal::net::tcp::TcpSocket**
Platform-independent implementation of the socket interface.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

7.623 src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::internal::net::tcp::TcpSocketInputStream**
Input stream for performing reads on a socket.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

7.624 src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

Data Structures

- class **decaf::internal::net::tcp::TcpSocketOutputStream**
Output stream for performing write operations on a socket.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

7.625 src/main/decaf/internal/net/URIEncoderDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <string>
```

Data Structures

- class **decaf::internal::net::URIEncoderDecoder**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.626 src/main/decaf/internal/net/URIHelper.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/internal/net/URIType.h>
```

Data Structures

- class **decaf::internal::net::URIHelper**

Helper class used by the URI classes in encoding and decoding of URI's.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.627 src/main/decaf/internal/net/URIType.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::net::URIType**

Basic type object that holds data that composes a given URI.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.628 src/main/decaf/internal/nio/BufferFactory.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/LongBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/ShortBuffer.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```


Data Structures

- class **decaf::internal::nio::BufferFactory**

*Factory class used by static methods in the **decaf::nio** (p. 132) package to create the various default version of the NIO interfaces.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.629 src/main/decaf/internal/nio/ByteBuffer.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/ShortBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/LongBuffer.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::ByteBuffer**

This class defines six categories of operations upon byte buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.630 `src/main/decaf/internal/nio/CharArrayBuffer.h` File Reference

```
#include <decaf/nio/CharBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::CharArrayBuffer**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.631 `src/main/decaf/internal/nio/DoubleArrayBuffer.h` File Reference

```
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
```

```
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::DoubleArrayBuffer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.632 src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference

```
#include <decaf/nio/FloatBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::FloatArrayBuffer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.633 src/main/decaf/internal/nio/IntArrayBuffer.h File Reference

```
#include <decaf/nio/IntBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::IntArrayBuffer**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.634 src/main/decaf/internal/nio/LongArrayBuffer.h File Reference

```
#include <decaf/nio/LongBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::LongArrayBuffer**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.635 src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference

```
#include <decaf/nio/ShortBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::ShortArrayBuffer**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.636 src/main/decaf/internal/security/unix/SecureRandomImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandomSpi.h>
```

Data Structures

- class **decaf::internal::security::SecureRandomImpl**

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::security**

7.637 src/main/decaf/internal/security/windows/SecureRandomImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandomSpi.h>
```

Data Structures

- class **decaf::internal::security::SecureRandomImpl**

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::security**

7.638 src/main/decaf/internal/util/ByteArrayAdapter.h File Reference

```
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
```

Data Structures

- class **decaf::internal::util::ByteArrayAdapter**
This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.
- union **decaf::internal::util::ByteArrayAdapter::Array**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.639 src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::util::concurrent::ConditionImpl**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.640 src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::util::concurrent::MutexImpl**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.641 src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::internal::util::concurrent::SynchronizableImpl**
A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.642 src/main/decaf/internal/util/concurrent/Transferer.h File Reference

```
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/util/concurrent/TimeoutException.h>
```

Data Structures

- class **decaf::internal::util::concurrent::Transferer< E >**
Shared internal API for dual stacks and queues.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.643 src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/locks/LockSupport.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/Thread.h>
```

Data Structures

- class **decaf::internal::util::concurrent::TransferQueue< E >**

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.644 src/main/decaf/internal/util/concurrent/TransferStack.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

Data Structures

- class **decaf::internal::util::concurrent::TransferStack< E >**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.645 src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::ConditionHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.646 src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::ConditionHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.647 src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::MutexHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.648 src/main/decaf/internal/util/concurrent/windows/MutexHandle.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::MutexHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.649 src/main/decaf/internal/util/GenericResource.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/internal/util/Resource.h>
```

Data Structures

- class **decaf::internal::util::GenericResource**< **T** >
*A Generic **Resource** (p. 3072) wraps some type and will delete it when the **Resource** (p. 3072) itself is deleted.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.650 src/main/decaf/internal/util/HexStringParser.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::internal::util::HexStringParser**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.651 src/main/decaf/internal/util/Resource.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::util::Resource**
Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.652 src/main/decaf/internal/util/TimerTaskHeap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/TimerTask.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::util::TimerTaskHeap**
A Binary Heap implemented specifically for the Timer class in Decaf Util.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.653 src/main/decaf/internal/util/zip/crc32.h File Reference

Variables

- local const unsigned long FAR **crc_table** [TBLS][256]

7.653.1 Variable Documentation

7.653.1.1 local const unsigned long FAR **crc_table**[TBLS][256]

7.654 src/main/decaf/internal/util/zip/deflate.h File Reference

```
#include "zutil.h"
```

Data Structures

- struct **ct_data_s**
- struct **tree_desc_s**
- struct **internal_state**

Defines

- #define **GZIP**
- #define **LENGTH_CODES** 29
- #define **LITERALS** 256
- #define **L_CODES** (LITERALS+1+LENGTH_CODES)
- #define **D_CODES** 30
- #define **BL_CODES** 19
- #define **HEAP_SIZE** (2*L_CODES+1)
- #define **MAX_BITS** 15
- #define **INIT_STATE** 42
- #define **EXTRA_STATE** 69
- #define **NAME_STATE** 73
- #define **COMMENT_STATE** 91
- #define **HCRC_STATE** 103
- #define **BUSY_STATE** 113
- #define **FINISH_STATE** 666
- #define **Freq** fc.freq
- #define **Code** fc.code
- #define **Dad** dl.dad
- #define **Len** dl.len
- #define **max_insert_length** max_lazy_match
- #define **put_byte**(s, c) {s->pending_buf[s->pending++] = (c);}
- #define **MIN_LOOKAHEAD** (MAX_MATCH+MIN_MATCH+1)
- #define **MAX_DIST**(s) ((s)->w_size-MIN_LOOKAHEAD)
- #define **WIN_INIT** MAX_MATCH
- #define **d_code**(dist) ((dist) < 256 ? __dist_code[dist] : __dist_code[256+((dist)>>7)])
- #define **_tr_tally_lit**(s, c, flush)
- #define **_tr_tally_dist**(s, distance, length, flush)

Typedefs

- typedef struct **ct_data_s** **ct_data**
- typedef struct static_tree_desc_s **static_tree_desc**
- typedef struct **tree_desc_s** **tree_desc**
- typedef **ush** **Pos**
- typedef **Pos** **FAR Posf**
- typedef unsigned **IPos**
- typedef struct **internal_state** **deflate_state**

Functions

- void ZLIB_INTERNAL _tr_init **OF** ((deflate_state *s))
- int ZLIB_INTERNAL _tr_tally **OF** ((deflate_state *s, unsigned dist, unsigned lc))
- void ZLIB_INTERNAL _tr_flush_block **OF** ((deflate_state *s, charf *buf, ulg stored_len, int last))

Variables

- uch ZLIB_INTERNAL _length_code []
- uch ZLIB_INTERNAL _dist_code []

7.654.1 Define Documentation

7.654.1.1 #define _tr_tally_dist(s, distance, length, flush)

Value:

```
{ uch len = (length); \
  ush dist = (distance); \
  s->d_buf[s->last_lit] = dist; \
  s->l_buf[s->last_lit++] = len; \
  dist--; \
  s->dyn_ltree[_length_code[len]+LITERALS+1].Freq++; \
  s->dyn_dtree[d_code(dist)].Freq++; \
  flush = (s->last_lit == s->lit_bufsize-1); \
}
```

7.654.1.2 #define _tr_tally_lit(s, c, flush)

Value:

```
{ uch cc = (c); \
  s->d_buf[s->last_lit] = 0; \
  s->l_buf[s->last_lit++] = cc; \
  s->dyn_ltree[cc].Freq++; \
  flush = (s->last_lit == s->lit_bufsize-1); \
}
```

```

7.654.1.3  #define BL_CODES 19

7.654.1.4  #define BUSY_STATE 113

7.654.1.5  #define Code fc.code

7.654.1.6  #define COMMENT_STATE 91

7.654.1.7  #define d_code( dist ) ((dist) < 256 ? _dist_code[dist] :
           _dist_code[256+((dist)>>7)])

7.654.1.8  #define D_CODES 30

7.654.1.9  #define Dad dl.dad

7.654.1.10 #define EXTRA_STATE 69

7.654.1.11 #define FINISH_STATE 666

7.654.1.12 #define Freq fc.freq

7.654.1.13 #define GZIP

7.654.1.14 #define HCRC_STATE 103

7.654.1.15 #define HEAP_SIZE (2*L_CODES+1)

7.654.1.16 #define INIT_STATE 42

7.654.1.17 #define L_CODES (LITERALS+1+LENGTH_CODES)

7.654.1.18 #define Len dl.len

7.654.1.19 #define LENGTH_CODES 29

7.654.1.20 #define LITERALS 256

7.654.1.21 #define MAX_BITS 15

7.654.1.22 #define MAX_DIST( s ) ((s)->w_size-MIN_LOOKAHEAD)

7.654.1.23 #define max_insert_length max_lazy_match

7.654.1.24 #define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)

7.654.1.25 #define NAME_STATE 73

7.654.1.26 #define put_byte( s, c ) {s->pending_buf[s->pending++] = (c);}

7.654.1.27 #define WIN_INIT MAX_MATCH

```

7.654.2 Typedef Documentation

```
7.654.2.1 typedef struct ct_data_s ct_data
```

```
7.654.2.2 typedef struct internal_state_s internal_state
```

```
7.654.2.3 typedef unsigned IPos
```

```
7.654.2.4 typedef unsigned Pos
```

```
7.654.2.5 typedef unsigned FAR Pos
```



```
#include "zlib.h"
#include <fcntl.h>
```

Data Structures

- struct **gz_state**

Defines

- #define **ZLIB_INTERNAL**
- #define **local** static
- #define **zstrerror()** "stdio error (consult errno)"
- #define **GZBUFSIZE** 8192
- #define **GZ_NONE** 0
- #define **GZ_READ** 7247
- #define **GZ_WRITE** 31153
- #define **GZ_APPEND** 1
- #define **LOOK** 0
- #define **COPY** 1
- #define **GZIP** 2
- #define **GT_OFF(x)** (sizeof(int) == sizeof(z_off64_t) && (x) > gz_intmax())

Typedefs

- typedef **gz_state FAR * gz_statep**

Functions

- **voidp** malloc **OF** ((**uInt** size))
- void free **OF** ((**voidpf** ptr))
- ZEXTERN **gzFile** ZEXPORT gzopen64 **OF** ((const char *, const char *))
- ZEXTERN **z_off64_t** ZEXPORT gzseek64 **OF** ((**gzFile**, **z_off64_t**, int))
- ZEXTERN **z_off64_t** ZEXPORT gztell64 **OF** ((**gzFile**))
- void ZLIB_INTERNAL gz_error **OF** ((**gz_statep**, int, const char *))
- unsigned ZLIB_INTERNAL gz_intmax **OF** ((void))

7.655.1 Define Documentation

7.655.1.1 `#define COPY 1`

7.655.1.2 `#define GT_OFF(x) (sizeof(int) == sizeof(z_off64_t) && (x) > gz_intmax())`

7.655.1.3 `#define GZ_APPEND 1`

7.655.1.4 `#define GZ_NONE 0`

7.655.1.5 `#define GZ_READ 7247`

7.655.1.6 `#define GZ_WRITE 31153`

7.655.1.7 `#define GZBUFSIZE 8192`

7.655.1.8 `#define GZIP 2`

7.655.1.9 `#define local static`

7.655.1.10 `#define LOOK 0`

7.655.1.11 `#define ZLIB_INTERNAL`

7.655.1.12 `#define zstrerror() "stdio error (consult errno)"`

7.655.2 Typedef Documentation

7.655.2.1 `typedef gz_state FAR* gz_statep`

7.655.3 Function Documentation

7.655.3.1 `voidp malloc OF ((uInt size))`

7.655.3.2 `unsigned ZLIB_INTERNAL gz_intmax OF ((void))`

7.655.3.3 `void ZLIB_INTERNAL gz_error OF ((gz_statep, int, const char *))`

7.655.3.4 `ZEXTERN z_off64_t ZEXPORT gztell64 OF ((gzFile))`

7.655.3.5 `ZEXTERN z_off64_t ZEXPORT gzseek64 OF ((gzFile, z_off64_t, int))`

7.655.3.6 `ZEXTERN gzFile ZEXPORT gzopen64 OF ((const char *, const char *))`

7.655.3.7 `void free OF ((voidpf ptr))`

7.656 src/main/decaf/internal/util/zip/inffast.h File Reference

Functions

-
- `void ZLIB_INTERNAL inflate_fast OF ((z_stream strm, unsigned start))`

7.656.1 Function Documentation

7.656.1.1 void ZLIB_INTERNAL inflate_fast OF ((z_stream strm, unsigned start))

7.657 src/main/decaf/internal/util/zip/inffixed.h File Reference

7.658 src/main/decaf/internal/util/zip/inflate.h File Reference

Data Structures

- struct inflate_state

Defines

- #define GUNZIP

Enumerations

- enum inflate_mode {
HEAD, FLAGS, TIME, OS,
EXLEN, EXTRA, NAME, COMMENT,
HCRC, DICTID, DICT, TYPE,
TYPEDO, STORED, COPY_, COPY,
TABLE, LENLENS, CODELENS, LEN_,
LEN, LENEXT, DIST, DISTEXT,
MATCH, LIT, CHECK, LENGTH,
DONE, BAD, MEM, SYNC }

7.658.1 Define Documentation

7.658.1.1 #define GUNZIP

7.658.2 Enumeration Type Documentation

7.658.2.1 enum inflate_mode

Enumerator:

HEAD

FLAGS

TIME

OS

EXLEN
EXTRA
NAME
COMMENT
HCRC
DICTID
DICT
TYPE
TYPEDO
STORED
COPY_
COPY
TABLE
LENLENS
CODELENS
LEN_
LEN
LENEXT
DIST
DISTEXT
MATCH
LIT
CHECK
LENGTH
DONE
BAD
MEM
SYNC

7.659 src/main/decaf/internal/util/zip/inftrees.h File Reference

Data Structures

- struct `code`

Defines

- `#define ENOUGH_LENS 852`
- `#define ENOUGH_DISTS 592`
- `#define ENOUGH (ENOUGH_LENS+ENOUGH_DISTS)`

Enumerations

- enum `codetype` { `CODES`, `LENS`, `DISTS` }

Functions

- int `ZLIB_INTERNAL inflate_table` OF ((`codetype` type, unsigned short FAR *`lens`, unsigned codes, `code` FAR *FAR *`table`, unsigned FAR *`bits`, unsigned short FAR *`work`))

7.659.1 Define Documentation

7.659.1.1 `#define ENOUGH (ENOUGH_LENS+ENOUGH_DISTS)`

7.659.1.2 `#define ENOUGH_DISTS 592`

7.659.1.3 `#define ENOUGH_LENS 852`

7.659.2 Enumeration Type Documentation

7.659.2.1 enum `codetype`

Enumerator:

`CODES`

`LENS`

`DISTS`

7.659.3 Function Documentation

7.659.3.1 int `ZLIB_INTERNAL inflate_table` OF ((`codetype` type, unsigned short FAR *`lens`, unsigned codes, `code` FAR *FAR *`table`, unsigned FAR *`bits`, unsigned short FAR *`work`))

7.660 src/main/decaf/internal/util/zip/trees.h File Reference

Variables

- local const `ct_data static_ltree` [L_CODES+2]
- local const `ct_data static_dtree` [D_CODES]
- const `uch ZLIB_INTERNAL_dist_code` [DIST_CODE_LEN]
- const `uch ZLIB_INTERNAL_length_code` [MAX_MATCH-MIN_MATCH+1]
- local const int `base_length` [LENGTH_CODES]
- local const int `base_dist` [D_CODES]

7.660.1 Variable Documentation

7.660.1.1 const `uch ZLIB_INTERNAL_dist_code`[DIST_CODE_LEN]

Initial value:

```
7.660.1.2  const uch ZLIB_INTERNAL _length_code[MAX_MATCH-MIN_
MATCH+1]
```

7.660.1.3 local const int base dist[D CODES]

$$\{ \begin{array}{cccccccccccc} 0, & 1, & 2, & 3, & 4, & 6, & 8, & 12, & 16, & 24, \\ 32, & 48, & 64, & 96, & 128, & 192, & 256, & 384, & 512, & 768, \\ 1024, & 1536, & 2048, & 3072, & 4096, & 6144, & 8192, & 12288, & 16384, & 24576 \end{array} \}$$

7.660.1.4 local const int base__length[LENGTH_CODES]**Initial value:**

```
{
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56,
64, 80, 96, 112, 128, 160, 192, 224, 0
}
```

7.660.1.5 local const ct_data static_dtree[D_CODES]**Initial value:**

```
{
{{ 0},{ 5}}, {{16},{ 5}}, {{ 8},{ 5}}, {{24},{ 5}}, {{ 4},{ 5}},
{{20},{ 5}}, {{12},{ 5}}, {{28},{ 5}}, {{ 2},{ 5}}, {{18},{ 5}},
{{10},{ 5}}, {{26},{ 5}}, {{ 6},{ 5}}, {{22},{ 5}}, {{14},{ 5}},
{{30},{ 5}}, {{ 1},{ 5}}, {{17},{ 5}}, {{ 9},{ 5}}, {{25},{ 5}},
{{ 5},{ 5}}, {{21},{ 5}}, {{13},{ 5}}, {{29},{ 5}}, {{ 3},{ 5}},
{{19},{ 5}}, {{11},{ 5}}, {{27},{ 5}}, {{ 7},{ 5}}, {{23},{ 5}}
}
```

7.660.1.6 local const ct_data static_ltree[L_CODES+2]**7.661 src/main/decaf/internal/util/zip/zconf.h File Reference**

```
#include <decaf/util/Config.h>
```

Defines

- **#define const**
- **#define MAX_MEM_LEVEL 9**
- **#define MAX_WBITS 15**
- **#define OF(args) ()**
- **#define ZEXTERN extern**
- **#define ZEXPORT**
- **#define ZEXPORTVA**
- **#define FAR**
- **#define SEEK_SET 0**
- **#define SEEK_CUR 1**
- **#define SEEK_END 2**
- **#define z_off_t long**
- **#define z_off64_t z_off_t**

Typedefs

- **typedef unsigned char Byte**
- **typedef unsigned int uInt**
- **typedef unsigned long uLong**

- typedef **Byte** FAR **Bytef**
- typedef char FAR **charf**
- typedef int FAR **intf**
- typedef **uInt** FAR **uIntf**
- typedef **uLong** FAR **uLongf**
- typedef **Byte** const * **voidpc**
- typedef **Byte** FAR * **voidpf**
- typedef **Byte** * **voidp**

7.661.1 Define Documentation

- 7.661.1.1 `#define const`
- 7.661.1.2 `#define FAR`
- 7.661.1.3 `#define MAX__MEM__LEVEL 9`
- 7.661.1.4 `#define MAX__WBITS 15`
- 7.661.1.5 `#define OF(args) ()`
- 7.661.1.6 `#define SEEK__CUR 1`
- 7.661.1.7 `#define SEEK__END 2`
- 7.661.1.8 `#define SEEK__SET 0`
- 7.661.1.9 `#define z__off64__t z__off__t`
- 7.661.1.10 `#define z__off__t long`
- 7.661.1.11 `#define ZEXPORT`
- 7.661.1.12 `#define ZEXPORTVA`
- 7.661.1.13 `#define ZEXTERN extern`

7.661.2 Typedef Documentation

- 7.661.2.1 `typedef unsigned char Byte`
- 7.661.2.2 `typedef Byte FAR Bytef`
- 7.661.2.3 `typedef char FAR charf`
- 7.661.2.4 `typedef int FAR intf`
- 7.661.2.5 `typedef unsigned int uInt`
- 7.661.2.6 `typedef uInt FAR uIntf`
- 7.661.2.7 `typedef unsigned long uLong`
- 7.661.2.8 `typedef uLong FAR uLongf`
- 7.661.2.9 `typedef Byte* voidp`
- 7.661.2.10 `typedef Byte const* voidpc`
- 7.661.2.11 `typedef Byte FAR* voidpf`

7.662 src/main/decaf/internal/util/zip/zlib.h File Reference

Generated on Sun Sep 11 2011 18:23:35 for activemq-cpp-3.2.1 by Doxygen

```
#include "zconf.h"
```

Data Structures

- struct **z_stream_s**
- struct **gz_header_s**
- struct **internal_state**

Defines

- #define **ZLIB_VERSION** "1.2.5"
- #define **ZLIB_VERNUM** 0x1250
- #define **ZLIB_VER_MAJOR** 1
- #define **ZLIB_VER_MINOR** 2
- #define **ZLIB_VER_REVISION** 5
- #define **ZLIB_VER_SUBREVISION** 0
- #define **Z_NO_FLUSH** 0
- #define **Z_PARTIAL_FLUSH** 1
- #define **Z_SYNC_FLUSH** 2
- #define **Z_FULL_FLUSH** 3
- #define **Z_FINISH** 4
- #define **Z_BLOCK** 5
- #define **Z_TREES** 6
- #define **Z_OK** 0
- #define **Z_STREAM_END** 1
- #define **Z_NEED_DICT** 2
- #define **Z_ERRNO** (-1)
- #define **Z_STREAM_ERROR** (-2)
- #define **Z_DATA_ERROR** (-3)
- #define **Z_MEM_ERROR** (-4)
- #define **Z_BUF_ERROR** (-5)
- #define **Z_VERSION_ERROR** (-6)
- #define **Z_NO_COMPRESSION** 0
- #define **Z_BEST_SPEED** 1
- #define **Z_BEST_COMPRESSION** 9
- #define **Z_DEFAULT_COMPRESSION** (-1)
- #define **Z_FILTERED** 1
- #define **Z_HUFFMAN_ONLY** 2
- #define **Z_RLE** 3
- #define **Z_FIXED** 4
- #define **Z_DEFAULT_STRATEGY** 0
- #define **Z_BINARY** 0
- #define **Z_TEXT** 1
- #define **Z_ASCII** Z_TEXT
- #define **Z_UNKNOWN** 2
- #define **Z_DEFLATED** 8
- #define **Z_NULL** 0
- #define **zlib_version** zlibVersion()
- #define **deflateInit**(strm, level) deflateInit_((strm), (level), ZLIB_VERSION, sizeof(z_stream))
- #define **inflateInit**(strm) inflateInit_((strm), ZLIB_VERSION, sizeof(z_stream))
- #define **deflateInit2**(strm, level, method, windowBits, memLevel, strategy)
- #define **inflateInit2**(strm, windowBits) inflateInit2_((strm), (windowBits), ZLIB_VERSION, sizeof(z_stream))
- #define **inflateBackInit**(strm, windowBits, window)

Typedefs

- typedef **voidpf** **alloc_func** **OF** ((**voidpf** opaque, **uInt** items, **uInt** size))
- typedef struct **z_stream_s** **z_stream**
- typedef **z_stream** FAR * **z_streamp**
- typedef struct **gz_header_s** **gz_header**
- typedef **gz_header** FAR * **gz_headerp**
- typedef **voidp** **gzFile**

Functions

- ZEXTERN const char *ZEXPORT zlibVersion **OF** ((void))
- ZEXTERN int ZEXPORT deflate **OF** ((**z_streamp** strm, int flush))
- ZEXTERN int ZEXPORT deflateEnd **OF** ((**z_streamp** strm))
- ZEXTERN int ZEXPORT deflateSetDictionary **OF** ((**z_streamp** strm, const **Bytef** *dictionary, **uInt** dictLength))
- ZEXTERN int ZEXPORT deflateCopy **OF** ((**z_streamp** dest, **z_streamp** source))
- ZEXTERN int ZEXPORT deflateParams **OF** ((**z_streamp** strm, int level, int strategy))
- ZEXTERN int ZEXPORT deflateTune **OF** ((**z_streamp** strm, int good_length, int max_lazy, int nice_length, int max_chain))
- ZEXTERN **uLong** ZEXPORT deflateBound **OF** ((**z_streamp** strm, **uLong** sourceLen))
- ZEXTERN int ZEXPORT deflatePrime **OF** ((**z_streamp** strm, int bits, int value))
- ZEXTERN int ZEXPORT deflateSetHeader **OF** ((**z_streamp** strm, **gz_headerp** head))
- ZEXTERN int ZEXPORT inflateReset2 **OF** ((**z_streamp** strm, int windowBits))
- ZEXTERN int ZEXPORT inflateBack **OF** ((**z_streamp** strm, in_func in, void FAR *in_desc, out_func out, void FAR *out_desc))
- ZEXTERN int ZEXPORT compress **OF** ((**Bytef** *dest, **uLongf** *destLen, const **Bytef** *source, **uLong** sourceLen))
- ZEXTERN int ZEXPORT compress2 **OF** ((**Bytef** *dest, **uLongf** *destLen, const **Bytef** *source, **uLong** sourceLen, int level))
- ZEXTERN **uLong** ZEXPORT compressBound **OF** ((**uLong** sourceLen))
- ZEXTERN **gzFile** ZEXPORT gzdopen **OF** ((int fd, const char *mode))
- ZEXTERN int ZEXPORT gzbuffer **OF** ((**gzFile** file, unsigned size))
- ZEXTERN int ZEXPORT gzsetparams **OF** ((**gzFile** file, int level, int strategy))
- ZEXTERN int ZEXPORT gzread **OF** ((**gzFile** file, **voidp** buf, unsigned len))
- ZEXTERN int ZEXPORT gzwrite **OF** ((**gzFile** file, **voidpc** buf, unsigned len))
- ZEXTERN int ZEXPORTVA gzprintf **OF** ((**gzFile** file, const char *format,...))
- ZEXTERN int ZEXPORT gzputs **OF** ((**gzFile** file, const char *s))
- ZEXTERN char *ZEXPORT gzgets **OF** ((**gzFile** file, char *buf, int len))
- ZEXTERN int ZEXPORT gzputc **OF** ((**gzFile** file, int c))
- ZEXTERN int ZEXPORT gzgetc **OF** ((**gzFile** file))
- ZEXTERN int ZEXPORT gzungetc **OF** ((int c, **gzFile** file))
- ZEXTERN int ZEXPORT gzflush **OF** ((**gzFile** file, int flush))
- ZEXTERN const char *ZEXPORT gzerror **OF** ((**gzFile** file, int *errnum))
- ZEXTERN **uLong** ZEXPORT Adler32 **OF** ((**uLong** Adler, const **Bytef** *buf, **uInt** len))
- ZEXTERN **uLong** ZEXPORT Crc32 **OF** ((**uLong** Crc, const **Bytef** *buf, **uInt** len))
- ZEXTERN int ZEXPORT deflateInit_ **OF** ((**z_streamp** strm, int level, const char *version, int stream_size))
- ZEXTERN int ZEXPORT inflateInit_ **OF** ((**z_streamp** strm, const char *version, int stream_size))

- ZEXTERN int ZEXPORT deflateInit2_ **OF** ((**z_streamp** strm, intlevel, intmethod, int windowBits, int memLevel, int strategy, const char *version, int stream_size))
- ZEXTERN int ZEXPORT inflateInit2_ **OF** ((**z_streamp** strm, intwindowBits, const char *version, int stream_size))
- ZEXTERN int ZEXPORT inflateBackInit_ **OF** ((**z_streamp** strm, int windowBits, unsigned char FAR *window, const char *version, int stream_size))
- ZEXTERN **gzFile** ZEXPORT gzopen **OF** ((const char *, const char *))
- ZEXTERN z_off_t ZEXPORT gzseek **OF** ((**gzFile**, z_off_t, int))
- ZEXTERN z_off_t ZEXPORT gztell **OF** ((**gzFile**))
- ZEXTERN uLong ZEXPORT Adler32_combine **OF** ((uLong, uLong, z_off_t))
- ZEXTERN const char *ZEXPORT zError **OF** ((int))
- ZEXTERN int ZEXPORT inflateSyncPoint **OF** ((**z_streamp**))
- ZEXTERN int ZEXPORT inflateUndermine **OF** ((**z_streamp**, int))

7.662.1 Define Documentation

7.662.1.1 `#define deflateInit(strm, level) deflateInit_((strm), (level),
ZLIB_VERSION, sizeof(z_stream))`

7.662.1.2 `#define deflateInit2(strm, level, method, windowBits, memLevel,
strategy)`

Value:

```
deflateInit2_((strm), (level), (method), (windowBits), (memLevel), \
              (strategy),          ZLIB_VERSION, sizeof(z_stream))
```

7.662.1.3 `#define inflateBackInit(strm, windowBits, window)`

Value:

```
inflateBackInit_((strm), (windowBits), (window), \
                  ZLIB_VERSION, sizeof(z_stream))
```

-
- 7.662.1.4 `#define inflateInit(strm) inflateInit_((strm), ZLIB_VERSION, sizeof(z_stream))`
- 7.662.1.5 `#define inflateInit2(strm, windowBits) inflateInit2_((strm), (windowBits), ZLIB_VERSION, sizeof(z_stream))`
- 7.662.1.6 `#define Z_ASCII Z_TEXT`
- 7.662.1.7 `#define Z_BEST_COMPRESSION 9`
- 7.662.1.8 `#define Z_BEST_SPEED 1`
- 7.662.1.9 `#define Z_BINARY 0`
- 7.662.1.10 `#define Z_BLOCK 5`
- 7.662.1.11 `#define Z_BUF_ERROR (-5)`
- 7.662.1.12 `#define Z_DATA_ERROR (-3)`
- 7.662.1.13 `#define Z_DEFAULT_COMPRESSION (-1)`
- 7.662.1.14 `#define Z_DEFAULT_STRATEGY 0`
- 7.662.1.15 `#define Z_DEFLATED 8`
- 7.662.1.16 `#define Z_ERRNO (-1)`
- 7.662.1.17 `#define Z_FILTERED 1`
- 7.662.1.18 `#define Z_FINISH 4`
- 7.662.1.19 `#define Z_FIXED 4`
- 7.662.1.20 `#define Z_FULL_FLUSH 3`
- 7.662.1.21 `#define Z_HUFFMAN_ONLY 2`
- 7.662.1.22 `#define Z_MEM_ERROR (-4)`
- 7.662.1.23 `#define Z_NEED_DICT 2`
- 7.662.1.24 `#define Z_NO_COMPRESSION 0`
- 7.662.1.25 `#define Z_NO_FLUSH 0`
- 7.662.1.26 `#define Z_NULL 0`
- 7.662.1.27 `#define Z_OK 0`
- 7.662.1.28 `#define Z_PARTIAL_FLUSH 1`
- 7.662.1.29 `#define Z_RLE 3`
- 7.662.1.30 `#define Z_STREAM_END 1`
-
- Generated on Sun Sep 11 2011 18:23:35 for activemq-cpp-3.2.1 by Doxygen
- 7.662.1.31 `#define Z_STREAM_ERROR (-2)`
- 7.662.1.32 `#define Z_SYNC_FLUSH 2`
- 7.662.1.33 `#define Z_TEXT 1`

Defines

- `#define ZLIB_INTERNAL`
- `#define local static`
- `#define ERR_MSG(err) z_errmsg[Z_NEED_DICT-(err)]`
- `#define ERR_RETURN(strm, err) return (strm->msg = (char*)ERR_MSG(err), (err))`
- `#define DEF_WBITS MAX_WBITS`
- `#define DEF_MEM_LEVEL 8`
- `#define STORED_BLOCK 0`
- `#define STATIC_TREES 1`
- `#define DYN_TREES 2`
- `#define MIN_MATCH 3`
- `#define MAX_MATCH 258`
- `#define PRESET_DICT 0x20`
- `#define OS_CODE 0x03`
- `#define F_OPEN(name, mode) fopen((name), (mode))`
- `#define Assert(cond, msg)`
- `#define Trace(x)`
- `#define Tracev(x)`
- `#define Tracevv(x)`
- `#define Tracec(c, x)`
- `#define Tracecv(c, x)`
- `#define ZALLOC(strm, items, size) (*((strm)->zalloc))((strm)->opaque, (items), (size))`
- `#define ZFREE(strm, addr) (*((strm)->zfree))((strm)->opaque, (voidpf)(addr))`
- `#define TRY_FREE(s, p) {if (p) ZFREE(s, p);}`

Typedefs

- `typedef unsigned char uch`
- `typedef uch FAR uchf`
- `typedef unsigned short ush`
- `typedef ush FAR ushf`
- `typedef unsigned long ulg`

Functions

- `ZEXTERN uLong ZEXPORT adler32_combine64 OF ((uLong, uLong, z_off_t))`
- `void ZLIB_INTERNAL zmemcpy OF ((Bytef *dest, const Bytef *source, uInt len))`
- `int ZLIB_INTERNAL zmemcmp OF ((const Bytef *s1, const Bytef *s2, uInt len))`
- `void ZLIB_INTERNAL zmemzero OF ((Bytef *dest, uInt len))`
- `voidpf ZLIB_INTERNAL zcalloc OF ((voidpf opaque, unsigned items, unsigned size))`
- `void ZLIB_INTERNAL zcfree OF ((voidpf opaque, voidpf ptr))`

Variables

- `const char *const z_errmsg [10]`

7.663.1 Define Documentation

- 7.663.1.1 `#define Assert(cond, msg)`
- 7.663.1.2 `#define DEF_MEM_LEVEL 8`
- 7.663.1.3 `#define DEF_WBITS MAX_WBITS`
- 7.663.1.4 `#define DYN_TREES 2`
- 7.663.1.5 `#define ERR_MSG(err) z_errmsg[Z_NEED_DICT-(err)]`
- 7.663.1.6 `#define ERR_RETURN(strm, err) return (strm->msg = (char*)ERR_MSG(err), (err))`
- 7.663.1.7 `#define F_OPEN(name, mode) fopen((name), (mode))`
- 7.663.1.8 `#define local static`
- 7.663.1.9 `#define MAX_MATCH 258`
- 7.663.1.10 `#define MIN_MATCH 3`
- 7.663.1.11 `#define OS_CODE 0x03`
- 7.663.1.12 `#define PRESET_DICT 0x20`
- 7.663.1.13 `#define STATIC_TREES 1`
- 7.663.1.14 `#define STORED_BLOCK 0`
- 7.663.1.15 `#define Trace(x)`
- 7.663.1.16 `#define Tracec(c, x)`
- 7.663.1.17 `#define Tracecv(c, x)`
- 7.663.1.18 `#define Tracev(x)`
- 7.663.1.19 `#define Tracevv(x)`
- 7.663.1.20 `#define TRY_FREE(s, p) {if (p) ZFREE(s, p);}`
- 7.663.1.21 `#define ZALLOC(strm, items, size) (*((strm)->zalloc))((strm)->opaque, (items), (size))`
- 7.663.1.22 `#define ZFREE(strm, addr) (*((strm)->zfree))((strm)->opaque, (voidpf)(addr))`
- 7.663.1.23 `#define ZLIB_INTERNAL`

7.663.2 Typedef Documentation

- 7.663.2.1 `typedef unsigned char uch`

Generated on Sun Sep 11 2011 18:23:35 for activemq-cpp-3.2.1 by Doxygen

7.663.2.2 `typedef uch FAR uchf`

- 7.663.2.3 `typedef unsigned long ulg`

- 7.663.2.4 `typedef unsigned short ush`

7.663.2.5 `typedef short FAR shf`

```
#include <vector>
```

Data Structures

- class **decaf::io::BlockingByteArrayInputStream**

This is a blocking version of a byte buffer stream.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.665 src/main/decaf/io/BufferedInputStream.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/io/FilterInputStream.h>
```

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::io::BufferedInputStream**

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.666 src/main/decaf/io/BufferedOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
```

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```


Data Structures

- class **decaf::io::BufferedOutputStream**

Wrapper around another output stream that buffers output before writing to the target output stream.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.667 src/main/decaf/io/ByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
```

Data Structures

- class **decaf::io::ByteArrayInputStream**

*A **ByteArrayInputStream** (p. 944) contains an internal buffer that contains bytes that may be read from the stream.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.668 src/main/decaf/io/ByteArrayOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <utility>
```

Data Structures

- class `decaf::io::ByteArrayOutputStream`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::io`

7.669 src/main/decaf/io/DataInput.h File Reference

```
#include <decaf/util/Config.h>
#include <vector>
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class `decaf::io::DataInput`
*The **DataInput** (p. 1452) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.*

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::io`

7.670 src/main/decaf/io/DataInputStream.h File Reference

```
#include <decaf/io/FilterInputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/io/UTFDataFormatException.h>
```

```
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::DataInputStream**

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.671 src/main/decaf/io/DataOutput.h File Reference

```
#include <decaf/util/Config.h>
#include <vector>
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::DataOutput**

*The **DataOutput** (p. 1468) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.672 src/main/decaf/io/DataOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <string>
```

Data Structures

- class **decaf::io::DataOutputStream**

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.673 src/main/decaf/io/EOFException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::EOFException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.674 src/main/decaf/io/FileDescriptor.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::io::FileDescriptor**

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.675 src/main/decaf/io/FilterInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::FilterInputStream**

*A **FilterInputStream** (p. 1771) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.676 src/main/decaf/io/FilterOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::FilterOutputStream**

This class is the superclass of all classes that filter output streams.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.677 src/main/decaf/io/Flushable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::Flushable**

*A **Flushable** (p. 1811) is a destination of data that can be flushed.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.678 src/main/decaf/io/InputStream.h File Reference

```
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::io::InputStream**

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.679 src/main/decaf/io/InputStreamReader.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/io/Reader.h>
```

Data Structures

- class **decaf::io::InputStreamReader**

*An **InputStreamReader** (p. 1919) is a bridge from byte streams to character streams.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.680 src/main/decaf/io/InterruptedIOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::InterruptedIOException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.681 src/main/decaf/io/IOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::io::IOException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.682 src/main/decaf/io/OutputStream.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/io/Flushable.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::OutputStream**

Base interface for any class that wants to represent an output stream of bytes.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.683 src/main/decaf/io/OutputStreamWriter.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/io/Writer.h>
```

Data Structures

- class **decaf::io::OutputStreamWriter**
A class for turning a character stream into a byte stream.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.684 src/main/decaf/io/PushbackInputStream.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/io/InputStream.h>
```

```
#include <decaf/io/FilterInputStream.h>
```

Data Structures

- class **decaf::io::PushbackInputStream**
*A **PushbackInputStream** (p. 2940) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.685 src/main/decaf/io/Reader.h File Reference

```
#include <string>
#include <decaf/lang/Readable.h>
#include <decaf/io/Closeable.h>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::Reader**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.686 src/main/decaf/io/UnsupportedEncodingException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::UnsupportedEncodingException**

Thrown when the the Character Encoding is not supported.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.687 src/main/decaf/io/UTFDataFormatException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::UTFDataFormatException**

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.688 src/main/decaf/io/Writer.h File Reference

```
#include <string>
#include <vector>
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/io/Flushable.h>
#include <decaf/lang/Appendable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::Writer**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.689 src/main/decaf/lang/Appendable.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Appendable**
An object to which char sequences and values can be appended.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.690 src/main/decaf/lang/ArrayPointer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/System.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/atomic/AtomicRefCounter.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
```

Data Structures

- class **decaf::lang::ArrayPointer< T, REFCOUNTER >**
*Decaf's implementation of a Smart **Pointer** (p. 2756) that is a template on a Type and is **Thread** (p. 3520) Safe if the default Reference Counter is used.*
- struct **decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayData**
- class **decaf::lang::ArrayPointerComparator< T, R >**
*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 669).*

- struct **std::less< decaf::lang::ArrayPointer< T > >**

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **std**

Functions

- template<typename T , typename R , typename U >
bool **decaf::lang::operator==** (const ArrayPointer< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
bool **decaf::lang::operator==** (const U *left, const ArrayPointer< T, R > &right)
- template<typename T , typename R , typename U >
bool **decaf::lang::operator!=** (const ArrayPointer< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
bool **decaf::lang::operator!=** (const U *left, const ArrayPointer< T, R > &right)

7.691 src/main/decaf/lang/Boolean.h File Reference

```
#include <string>
#include <decaf/lang/Comparable.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Boolean**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.692 src/main/decaf/lang/Byte.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Byte**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.693 src/main/decaf/lang/Character.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

Data Structures

- class **decaf::lang::Character**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.694 src/main/decaf/lang/CharSequence.h File Reference

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::CharSequence**

*A **CharSequence** (p. 1053) is a readable sequence of char values.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.695 src/main/decaf/lang/Comparable.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Comparable< T >**

This interface imposes a total ordering on the objects of each class that implements it.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.696 src/main/decaf/lang/Double.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/Comparable.h>
```

```
#include <decaf/lang/Number.h>
```

```
#include <decaf/lang/exceptions/NumberFormatException.h>
```

```
#include <string>
```

Data Structures

- class **decaf::lang::Double**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.697 src/main/decaf/lang/Exception.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/lang/exceptions/ExceptionDefines.h>
#include <decaf/util/Config.h>
#include <stdarg.h>
#include <sstream>
```

Data Structures

- class **decaf::lang::Exception**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.698 src/main/decaf/lang/exceptions/ClassCastException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::ClassCastException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.699 src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class `decaf::lang::exceptions::IllegalArgumentException`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::lang`
- namespace `decaf::lang::exceptions`

7.700 src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class `decaf::lang::exceptions::IllegalMonitorStateException`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::lang`
- namespace `decaf::lang::exceptions`

7.701 src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class `decaf::lang::exceptions::IllegalThreadStateException`

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.702 src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IndexOutOfBoundsException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.703 src/main/decaf/lang/exceptions/InterruptedException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::InterruptedException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.704 src/main/decaf/lang/exceptions/InvalidStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::InvalidStateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.705 src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NoSuchElementException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.706 src/main/decaf/lang/exceptions/NullPointerException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NullPointerException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.707 src/main/decaf/lang/exceptions/NumberFormatException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NumberFormatException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.708 src/main/decaf/lang/exceptions/RuntimeException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::RuntimeException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.709 src/main/decaf/lang/Float.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Float**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.710 src/main/decaf/lang/Integer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
#include <decaf/lang/exceptions/NumberFormatException.h>
```

Data Structures

- class **decaf::lang::Integer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.711 src/main/decaf/lang/Iterable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
```

Data Structures

- class **decaf::lang::Iterable**< E >

*Implementing this interface allows an object to be cast to an **Iterable** (p. 2011) type for generic collections API calls.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.712 src/main/decaf/lang/Long.h File Reference

```
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Long**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.713 src/main/decaf/lang/Math.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Math**

*The class **Math** (p. 2339) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.714 src/main/decaf/lang/Number.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Number**

*The abstract class **Number** (p. 2653) is the superclass of classes **Byte** (p. 884), **Double** (p. 1672), **Float** (p. 1780), **Integer** (p. 1941), **Long** (p. 2267), and **Short** (p. 3220).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.715 src/main/decaf/lang/Pointer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/ClassCastException.h>
#include <decaf/util/concurrent/atomic/AtomicRefCounter.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
```

Data Structures

- struct **decaf::lang::STATIC_CAST_TOKEN**
- struct **decaf::lang::DYNAMIC_CAST_TOKEN**
- class **decaf::lang::Pointer< T, REFCOUNTER >**

*Decaf's implementation of a Smart **Pointer** (p. 2756) that is a template on a Type and is **Thread** (p. 3520) Safe if the default Reference Counter is used.*

- class **decaf::lang::PointerComparator**< T, R >

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2756) instance.*

- struct **std::less**< **decaf::lang::Pointer**< T > >

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **std**

Functions

- template<typename T , typename R , typename U >
bool **decaf::lang::operator==** (const Pointer< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
bool **decaf::lang::operator==** (const U *left, const Pointer< T, R > &right)
- template<typename T , typename R , typename U >
bool **decaf::lang::operator!=** (const Pointer< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
bool **decaf::lang::operator!=** (const U *left, const Pointer< T, R > &right)

7.716 src/main/decaf/lang/Readable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::lang::Readable**

*A **Readable** (p. 2958) is a source of characters.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**
- namespace **decaf::lang**

7.717 src/main/decaf/lang/Runnable.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Runnable**

Interface for a runnable object - defines a task that can be run by a thread.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.718 src/main/decaf/lang/Runtime.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Runtime**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.719 src/main/decaf/lang/Short.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Short**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.720 src/main/decaf/lang/String.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

Data Structures

- class **decaf::lang::String**
*The **String** (p. 3427) class represents an immutable sequence of chars.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.721 src/main/decaf/lang/System.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/util/Map.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/internal/AprPool.h>
#include <string>
```

Data Structures

- class **decaf::lang::System**

The **System** (p. 3487) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.722 src/main/decaf/lang/Thread.h File Reference

```
#include <decaf/lang/exceptions/IllegalThreadStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Thread**

A **Thread** (p. 3520) is a concurrent unit of execution.

- class **decaf::lang::Thread::UncaughtExceptionHandler**

Interface for handlers invoked when a **Thread** (p. 3520) abruptly terminates due to an uncaught exception.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**
- namespace **decaf::lang**

7.723 src/main/decaf/lang/ThreadGroup.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::ThreadGroup**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.724 src/main/decaf/lang/Throwable.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Throwable**
This class represents an error that has occurred.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.725 src/main/decaf/net/BindException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::BindException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.726 src/main/decaf/net/ConnectException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::ConnectException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.727 src/main/decaf/net/HttpRetryException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class `decaf::net::HttpRetryException`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::net`

7.728 src/main/decaf/net/Inet4Address.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

Data Structures

- class `decaf::net::Inet4Address`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::net`

7.729 src/main/decaf/net/Inet6Address.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

Data Structures

- class `decaf::net::Inet6Address`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::net`

7.730 src/main/decaf/net/InetAddress.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/ArrayPointer.h>
```

Data Structures

- class **decaf::net::InetAddress**
Represents an IP address.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.731 src/main/decaf/net/InetSocketAddress.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketAddress.h>
#include <string>
```

Data Structures

- class **decaf::net::InetSocketAddress**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.732 src/main/decaf/net/MalformedURLException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class `decaf::net::MalformedURLException`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::net`

7.733 `src/main/decaf/net/NoRouteToHostException.h` File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/net/SocketException.h>
```

Data Structures

- class `decaf::net::NoRouteToHostException`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::net`

7.734 `src/main/decaf/net/PortUnreachableException.h` File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/net/SocketException.h>
```

Data Structures

- class `decaf::net::PortUnreachableException`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.735 src/main/decaf/net/ProtocolException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::ProtocolException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.736 src/main/decaf/net/ServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
#include <decaf/net/SocketImpl.h>
#include <decaf/net/SocketImplFactory.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/UnknownHostException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/io/IOException.h>
#include <string>
```

Data Structures

- class **decaf::net::ServerSocket**

This class implements server sockets.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.737 src/main/decaf/net/ServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

Data Structures

- class **decaf::net::ServerSocketFactory**

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.738 src/main/decaf/net/Socket.h File Reference

```
#include <decaf/net/InetAddress.h>
#include <decaf/net/SocketImplFactory.h>
#include <decaf/net/SocketException.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/UnknownHostException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::Socket**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.739 src/main/decaf/net/SocketAddress.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketAddress**

*Base class for protocol specific **Socket** (p. 3281) addresses.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.740 src/main/decaf/net/SocketError.h File Reference

```
#include <string>
```

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketError**

Static utility class to simplify handling of error codes for socket operations.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.741 src/main/decaf/net/SocketException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::SocketException**
Exception for errors when manipulating sockets.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.742 src/main/decaf/net/SocketFactory.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/io/IOException.h>  
#include <decaf/net/UnknownHostException.h>
```

Data Structures

- class **decaf::net::SocketFactory**
*The **SocketFactory** (p. 3301) is used to create **Socket** (p. 3281) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.743 src/main/decaf/net/SocketImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/FileDescriptor.h>
#include <decaf/net/SocketException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/net/SocketOptions.h>
#include <string>
```

Data Structures

- class **decaf::net::SocketImpl**
*Acts as a base class for all physical **Socket** (p. 3281) implementations.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.744 src/main/decaf/net/SocketImplFactory.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketImplFactory**
*Factory class interface for a Factory that creates *SocketImpl* objects.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.745 src/main/decaf/net/SocketOptions.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class `decaf::net::SocketOptions`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::net`

7.746 src/main/decaf/net/SocketTimeoutException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/io/InterruptedIOException.h>
```

Data Structures

- class `decaf::net::SocketTimeoutException`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::net`

7.747 src/main/decaf/net/ssl/SSLContext.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/net/ssl/SSLContextSpi.h>
```

Data Structures

- class `decaf::net::ssl::SSLContext`
*Represents an implementation of the Secure **Socket** (p. 3281) Layer for streaming based sockets.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.748 src/main/decaf/net/ssl/SSLContextSpi.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandom.h>
```

Data Structures

- class **decaf::net::ssl::SSLContextSpi**

*Defines the interface that should be provided by an **SSLContext** (p. 3321) provider.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.749 src/main/decaf/net/ssl/SSLParameters.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
#include <vector>
```

Data Structures

- class **decaf::net::ssl::SSLParameters**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.750 src/main/decaf/net/ssl/SSLServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocket.h>
```

Data Structures

- class **decaf::net::ssl::SSLServerSocket**

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.751 src/main/decaf/net/ssl/SSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocketFactory.h>
#include <vector>
#include <string>
```

Data Structures

- class **decaf::net::ssl::SSLServerSocketFactory**

Factory class interface that provides methods to create SSL Server Sockets.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.752 src/main/decaf/net/ssl/SSLSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/Socket.h>
#include <decaf/net/ssl/SSLParameters.h>
```

Data Structures

- class **decaf::net::ssl::SSLSocket**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.753 src/main/decaf/net/ssl/SSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketFactory.h>
#include <vector>
#include <string>
```

Data Structures

- class **decaf::net::ssl::SSLSocketFactory**

*Factory class interface for a **SocketFactory** (p. 3301) that can create **SSLSocket** (p. 3337) objects.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.754 src/main/decaf/net/UnknownHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::UnknownHostException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.755 src/main/decaf/net/UnknownServiceException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::UnknownServiceException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.756 src/main/decaf/net/URI.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/net/MalformedURLException.h>
#include <decaf/net/URL.h>
#include <decaf/internal/net/URIType.h>
#include <string>
```

Data Structures

- class **decaf::net::URI**

*This class represents an instance of a **URI** (p. 3660) as defined by RFC 2396.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.757 src/main/decaf/net/URISyntaxException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::net::URISyntaxException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.758 src/main/decaf/net/URL.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::net::URL**

*Class **URL** (p. 3697) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.759 src/main/decaf/net/URLDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::net::URLDecoder**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.760 src/main/decaf/net/URLEncoder.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::net::URLEncoder**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.761 src/main/decaf/nio/Buffer.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

```
#include <decaf/nio/InvalidMarkException.h>
```

Data Structures

- class **decaf::nio::Buffer**

A container for data of a specific primitive type.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.762 src/main/decaf/nio/BufferOverflowException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::nio::BufferOverflowException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.763 src/main/decaf/nio/BufferUnderflowException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::nio::BufferUnderflowException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.764 src/main/decaf/nio/ByteBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::ByteBuffer**

This class defines six categories of operations upon byte buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.765 src/main/decaf/nio/CharBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Appendable.h>
```

Data Structures

- class **decaf::nio::CharBuffer**

This class defines four categories of operations upon character buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.766 src/main/decaf/nio/DoubleBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::DoubleBuffer**

This class defines four categories of operations upon double buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.767 src/main/decaf/nio/FloatBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::FloatBuffer**

This class defines four categories of operations upon float buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.768 src/main/decaf/nio/IntBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```


Data Structures

- class **decaf::nio::IntBuffer**

This class defines four categories of operations upon int buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.769 src/main/decaf/nio/InvalidMarkException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **decaf::nio::InvalidMarkException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.770 src/main/decaf/nio/LongBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::LongBuffer**

This class defines four categories of operations upon long long buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.771 src/main/decaf/nio/ReadOnlyBufferException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **decaf::nio::ReadOnlyBufferException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.772 src/main/decaf/nio/ShortBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
```

```
#include <decaf/lang/Comparable.h>
```

```
#include <decaf/lang/exceptions/NullPointerException.h>
```

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

```
#include <decaf/nio/BufferUnderflowException.h>
```

```
#include <decaf/nio/BufferOverflowException.h>
```

```
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class `decaf::nio::ShortBuffer`

This class defines four categories of operations upon short buffers:

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::nio`

7.773 src/main/decaf/security/auth/x500/X500Principal.h File Reference

```
#include <string>
#include <vector>
#include <decaf/security/Principal.h>
#include <decaf/util/Map.h>
#include <decaf/io/InputStream.h>
```

Data Structures

- class `decaf::security::auth::x500::X500Principal`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::security`
- namespace `decaf::security::auth`
- namespace `decaf::security::auth::x500`

7.774 src/main/decaf/security/cert/Certificate.h File Reference

```
#include <vector>
#include <decaf/util/Config.h>
#include <decaf/security/InvalidKeyException.h>
#include <decaf/security/NoSuchAlgorithmException.h>
```

```
#include <decaf/security/SignatureException.h>
#include <decaf/security/cert/CertificateEncodingException.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::Certificate**
Base interface for all identity certificates.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.775 src/main/decaf/security/cert/CertificateEncodingException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateEncodingException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.776 src/main/decaf/security/cert/CertificateException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class `decaf::security::cert::CertificateException`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::security`
- namespace `decaf::security::cert`

7.777 src/main/decaf/security/cert/CertificateExpiredException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class `decaf::security::cert::CertificateExpiredException`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::security`
- namespace `decaf::security::cert`

7.778 src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class `decaf::security::cert::CertificateNotYetValidException`

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.779 src/main/decaf/security/cert/CertificateParsingException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateParsingException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.780 src/main/decaf/security/cert/X509Certificate.h File Reference

```
#include <decaf/security/cert/Certificate.h>
```

```
#include <decaf/util/Config.h>
```

```
#include <decaf/util/Date.h>
```

Data Structures

- class **decaf::security::cert::X509Certificate**

Base interface for all identity certificates.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.781 src/main/decaf/security/GeneralSecurityException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::security::GeneralSecurityException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.782 src/main/decaf/security/InvalidKeyException.h File Reference

```
#include <decaf/security/KeyException.h>
```

Data Structures

- class **decaf::security::InvalidKeyException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.783 src/main/decaf/security/Key.h File Reference

```
#include <vector>
```

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::Key**

*The **Key** (p. 2149) interface is the top-level interface for all keys.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.784 src/main/decaf/security/KeyException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::KeyException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.785 src/main/decaf/security/KeyManagementException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/KeyException.h>
```

Data Structures

- class **decaf::security::KeyManagementException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.786 src/main/decaf/security/NoSuchAlgorithmException.h File Reference

```
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::NoSuchAlgorithmException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.787 src/main/decaf/security/NoSuchProviderException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::NoSuchProviderException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.788 src/main/decaf/security/Principal.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::Principal**

Base interface for a principal, which can represent an individual or organization.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.789 src/main/decaf/security/PublicKey.h File Reference

```
#include <decaf/security/Key.h>
```

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::PublicKey**

A public key.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.790 src/main/decaf/security/SecureRandom.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/util/Random.h>
```

```
#include <decaf/security/SecureRandomSpi.h>
```

```
#include <memory>
```

Data Structures

- class **decaf::security::SecureRandom**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.791 src/main/decaf/security/SecureRandomSpi.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::SecureRandomSpi**

Interface class used by Security Service Providers to implement a source of secure random bytes.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.792 src/main/decaf/security/SignatureException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::SignatureException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.793 src/main/decaf/util/AbstractCollection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Collection.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractCollection**< E >

*This class provides a skeletal implementation of the **Collection** (p. 1097) interface, to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.794 src/main/decaf/util/AbstractList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/List.h>
#include <memory>
```

Data Structures

- class `decaf::util::AbstractList< E >`

*This class provides a skeletal implementation of the **List** (p. 2190) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`

7.795 src/main/decaf/util/AbstractMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Map.h>
#include <decaf/util/Set.h>
#include <memory>
```

Data Structures

- class `decaf::util::AbstractMap< K, V, COMPARATOR >`

*This class provides a skeletal implementation of the **Map** (p. 2305) interface, to minimize the effort required to implement this interface.*

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`

7.796 src/main/decaf/util/AbstractQueue.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Queue.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractQueue**< E >

*This class provides skeletal implementations of some **Queue** (p. 2948) operations.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.797 src/main/decaf/util/AbstractSequentialList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractList.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractSequentialList**< E >

*This class provides a skeletal implementation of the **List** (p. 2190) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.798 src/main/decaf/util/AbstractSet.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Set.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractSet< E >**

*This class provides a skeletal implementation of the **Set** (p. 3220) interface to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.799 src/main/decaf/util/Collection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Synchronizable.h>
```

Data Structures

- class **decaf::util::Collection**< **E** >
The root interface in the collection hierarchy.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.800 src/main/decaf/util/Comparator.h File Reference

```
#include <decaf/util/Config.h>
#include <algorithm>
#include <functional>
```

Data Structures

- class **decaf::util::Comparator**< **T** >
A comparison function, which imposes a total ordering on some collection of objects.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.801 src/main/decaf/util/comparators/Less.h File Reference

```
#include <decaf/util/Comparator.h>
```

Data Structures

- class **decaf::util::comparators::Less**< **E** >
*Simple **Less** (p. 2182) **Comparator** (p. 1127) that compares to elements to determine if the first is less than the second.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::comparators**

7.802 src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicBoolean**

A boolean value that may be updated atomically.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.803 src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <string>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicInteger**

An int value that may be updated atomically.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.804 src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h File Reference

```
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicRefCounter**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.805 src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/Long.h>
```

```
#include <apr_atomic.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicReference< T >**

An Pointer reference that may be updated atomically.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.806 src/main/decaf/util/concurrent/BlockingQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

Data Structures

- class **decaf::util::concurrent::BlockingQueue< E >**

*A **decaf::util::Queue** (p. 2948) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.807 src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::BrokenBarrierException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.808 src/main/decaf/util/concurrent/Callable.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::Callable< V >**
A task that returns a result and may throw an exception.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.809 src/main/decaf/util/concurrent/CancellationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::CancellationException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.810 src/main/decaf/util/concurrent/Concurrent.h File Reference

```
#include <decaf/util/concurrent/Lock.h>
```

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

Defines

- `#define WAIT_INFINITE 0xFFFFFFFF`

The synchronized macro defines a mechanism for synchronizing a section of code.

- `#define synchronized(W)`

7.810.1 Define Documentation

7.810.1.1 `#define synchronized(W)`

Value:

```
if(false){}
else
    for( decaf::util::concurrent::Lock lock_W(W);
        lock_W.isLocked(); lock_W.unlock() )
```

7.810.1.2 `#define WAIT_INFINITE 0xFFFFFFFF`

The synchronized macro defines a mechanism for synchronizing a section of code.

The macro must be passed an object that implements the Synchronizable interface.

The macro works by creating a for loop that will loop exactly once, creating a Lock object that is scoped to the loop. Once the loop completes and exits the Lock object goes out of scope releasing the lock on object W. For added safety the if else is used because not all compilers restrict the lifetime of loop variables to the loop, they will however restrict them to the scope of the else.

The macro would be used as follows.

Synchronizable X;

```
somefunction() { synchronized(X) (p. 4287) { // Do something that needs synchronizing. } }
```

7.811 src/main/decaf/util/concurrent/ConcurrentMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **decaf::util::concurrent::ConcurrentMap**< K, V, COMPARATOR >
*Interface for a **Map** (p. 2305) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2305) interface.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.812 src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/ConcurrentMap.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

Data Structures

- class **decaf::util::concurrent::ConcurrentStlMap**< K, V, COMPARATOR >
***Map** (p. 2305) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.813 src/main/decaf/util/concurrent/CountDownLatch.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::CountDownLatch**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.814 src/main/decaf/util/concurrent/Delayed.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/TimeUnit.h>
```

Data Structures

- class **decaf::util::concurrent::Delayed**

A mix-in style interface for marking objects that should be acted upon after a given delay.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.815 src/main/decaf/util/concurrent/ExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::ExecutionException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.816 src/main/decaf/util/concurrent/Executor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

Data Structures

- class **decaf::util::concurrent::Executor**

*An object that executes submitted **decaf.lang.Runnable** (p. 3111) tasks.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.817 src/main/decaf/util/concurrent/ExecutorService.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Executor.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

Data Structures

- class **decaf::util::concurrent::ExecutorService**
*An **Executor** (p. 1749) that provides methods to manage termination and methods that can produce a **Future** (p. 1840) for tracking progress of one or more asynchronous tasks.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.818 src/main/decaf/util/concurrent/Future.h File Reference

Data Structures

- class **decaf::util::concurrent::Future< V >**
*A **Future** (p. 1840) represents the result of an asynchronous computation.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.819 src/main/decaf/util/concurrent/Lock.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Lock**

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.820 src/main/decaf/util/concurrent/locks/Lock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/util/concurrent/locks/Condition.h>
```

Data Structures

- class **decaf::util::concurrent::locks::Lock**

***Lock** (p. 2229) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.821 src/main/decaf/util/concurrent/locks/Condition.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
```

Data Structures

- class **decaf::util::concurrent::locks::Condition**

Condition (p. 1156) factors out the *Mutex* (p. 2604) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary *Lock* (p. 2229) implementations.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.822 src/main/decaf/util/concurrent/locks/LockSupport.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::locks::LockSupport**

Basic thread blocking primitives for creating locks and other synchronization classes.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.823 src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference

Data Structures

- class **decaf::util::concurrent::locks::ReadWriteLock**
*A **ReadWriteLock** (p. 2968) maintains a pair of associated locks, one for read-only operations and one for writing.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.824 src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/locks/Lock.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::util::concurrent::locks::ReentrantLock**
*A reentrant mutual exclusion **Lock** (p. 2229) with extended capabilities.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.825 src/main/decaf/util/concurrent/Mutex.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/lang/Thread.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Mutex**

***Mutex** (p. 2604) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.826 src/main/decaf/util/concurrent/PooledThread.h File Reference

```
#include <decaf/lang/Thread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::PooledThread**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.827 src/main/decaf/util/concurrent/PooledThreadListener.h File Reference

```
#include <decaf/lang/Exception.h>
```

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::PooledThreadListener**
Abstract Listener Interface for users of `ThreadPool` (p. 3531).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.828 src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::RejectedExecutionException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.829 src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/Runnable.h>
```

```
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

Data Structures

- class **decaf::util::concurrent::RejectedExecutionHandler**
A handler for tasks that cannot be executed by a `ThreadPoolExecutor` (p. ??).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.830 src/main/decaf/util/concurrent/Semaphore.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <memory>
```

Data Structures

- class **decaf::util::concurrent::Semaphore**
A counting semaphore.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.831 src/main/decaf/util/concurrent/Synchronizable.h File Reference

```
#include <decaf/lang/exceptions/RuntimeException.h>
```

```
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Synchronizable**

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.832 src/main/decaf/util/concurrent/SynchronousQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/BlockingQueue.h>
#include <vector>
```

Data Structures

- class **decaf::util::concurrent::SynchronousQueue< E >**

*A **blocking queue** (p. 774) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.*

- class **decaf::util::concurrent::SynchronousQueue< E >::EmptyIterator**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.833 src/main/decaf/util/concurrent/TaskListener.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::TaskListener**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.834 src/main/decaf/util/concurrent/ThreadFactory.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::ThreadFactory**
*public interface **ThreadFactory** (p. 3529)*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.835 src/main/decaf/util/concurrent/ThreadPool.h File Reference

```
#include <decaf/lang/Runnable.h>
```

```
#include <decaf/util/concurrent/PooledThread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/concurrent/TaskListener.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <vector>
```

Data Structures

- class **decaf::util::concurrent::ThreadPool**

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.836 src/main/decaf/util/concurrent/TimeoutException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::TimeoutException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.837 src/main/decaf/util/concurrent/TimeUnit.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::util::concurrent::TimeUnit**

*A **TimeUnit** (p. 3559) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.838 src/main/decaf/util/Date.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

Data Structures

- class **decaf::util::Date**

Wrapper class around a time value in milliseconds.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.839 src/main/decaf/util/Iterator.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **decaf::util::Iterator**< T >

Defines an object that can be used to iterate over the elements of a collection.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.840 src/main/decaf/util/List.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/util/ListIterator.h>
```

Data Structures

- class **decaf::util::List**< E >

An ordered collection (also known as a sequence).

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.841 src/main/decaf/util/ListIterator.h File Reference

```
#include <decaf/util/Iterator.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::ListIterator< E >**

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.842 src/main/decaf/util/logging/ConsoleHandler.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/StreamHandler.h>
#include <decaf/util/logging/SimpleFormatter.h>
#include <decaf/io/IOException.h>
#include <decaf/internal/io/StandardErrorOutputStream.h>
```

Data Structures

- class **decaf::util::logging::ConsoleHandler**

*This **Handler** (p. 1852) publishes log records to *System.err*.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.843 src/main/decaf/util/logging/EventManager.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
```

Data Structures

- class **decaf::util::logging::EventManager**

***ErrorManager** (p. 1710) objects can be attached to *Handlers* to process any error that occur on a *Handler* (p. 1852) during Logging.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.844 src/main/decaf/util/logging/Filter.h File Reference

```
#include <decaf/util/logging/LogRecord.h>
```

Data Structures

- class **decaf::util::logging::Filter**

*A **Filter** (p. 1770) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.845 src/main/decaf/util/logging/Formatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Handler.h>
```

Data Structures

- class **decaf::util::logging::Formatter**

*A **Formatter** (p. 1838) provides support for formatting **LogRecords**.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.846 src/main/decaf/util/logging/Handler.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/util/logging/Level.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <string>
```

Data Structures

- class **decaf::util::logging::Handler**

*A **Handler** (p. 1852) object takes log messages from a **Logger** (p. 2237) and exports them.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.847 src/main/decaf/util/logging/Level.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::logging::Level**
*The **Level** (p. 2185) class defines a set of standard logging levels that can be used to control logging output.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.848 src/main/decaf/util/logging/Logger.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/util/logging/LogManager.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <list>
#include <string>
#include <stdarg.h>
```


Data Structures

- class `decaf::util::logging::Logger`

*A **Logger** (p. 2237) object is used to log messages for a specific system or application component.*

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`
- namespace `decaf::util::logging`

7.849 src/main/decaf/util/logging/LoggerCommon.h File Reference

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`
- namespace `decaf::util::logging`

Enumerations

- enum `decaf::util::logging::Levels` {
 `decaf::util::logging::Off`, `decaf::util::logging::Null`, `decaf::util::logging::Markblock`,
 `decaf::util::logging::Debug`,
 `decaf::util::logging::Info`, `decaf::util::logging::Warn`, `decaf::util::logging::Error`,
 `decaf::util::logging::Fatal`,
 `decaf::util::logging::Throwing` }

Defines an enumeration for logging levels.

7.850 src/main/decaf/util/logging/LoggerDefines.h File Reference

```
#include <decaf/util/logging/SimpleLogger.h>
#include <sstream>
```

Defines

- `#define LOGDECAF_DECLARE(loggerName) static decaf::util::logging::SimpleLogger loggerName;`
- `#define LOGDECAF_INITIALIZE(loggerName, className, loggerFamily) decaf::util::logging::SimpleLogger className::loggerName(loggerFamily);`
- `#define LOGDECAF_DECLARE_LOCAL(loggerName) decaf::util::logging::Logger loggerName;`
- `#define LOGDECAF_DEBUG(logger, message) logger.debug(__FILE__, __LINE__, message);`
- `#define LOGDECAF_DEBUG_1(logger, message, value)`
- `#define LOGDECAF_INFO(logger, message) logger.info(__FILE__, __LINE__, message);`
- `#define LOGDECAF_ERROR(logger, message) logger.error(__FILE__, __LINE__, message);`
- `#define LOGDECAF_WARN(logger, message) logger.warn(__FILE__, __LINE__, message);`
- `#define LOGDECAF_FATAL(logger, message) logger.fatal(__FILE__, __LINE__, message);`

7.850.1 Define Documentation

7.850.1.1 `#define LOGDECAF_DEBUG(logger, message) logger.debug(__FILE__, __LINE__, message);`

7.850.1.2 `#define LOGDECAF_DEBUG_1(logger, message, value)`

Value:

```

;          \
{          \
    std::ostringstream ostream;          \
    ostream << message << value;          \
    logger.debug(__FILE__, __LINE__, ostream.str());          \
}

```

- 7.850.1.3 `#define LOGDECAF_DECLARE(loggerName) static
decaf::util::logging::SimpleLogger loggerName;`
- 7.850.1.4 `#define LOGDECAF_DECLARE_LOCAL(loggerName
) decaf::util::logging::Logger loggerName;`
- 7.850.1.5 `#define LOGDECAF_ERROR(logger, message
) logger.error(__FILE__, __LINE__, message);`
- 7.850.1.6 `#define LOGDECAF_FATAL(logger, message
) logger.fatal(__FILE__, __LINE__, message);`
- 7.850.1.7 `#define LOGDECAF_INFO(logger, message
) logger.info(__FILE__, __LINE__, message);`
- 7.850.1.8 `#define LOGDECAF_INITIALIZE(loggerName,
className, loggerFamily) decaf::util::logging::SimpleLogger
className::loggerName(loggerFamily);`
- 7.850.1.9 `#define LOGDECAF_WARN(logger, message
) logger.warn(__FILE__, __LINE__, message);`

7.851 src/main/decaf/util/logging/LoggerHierarchy.h File Reference

Data Structures

- class `decaf::util::logging::LoggerHierarchy`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`
- namespace `decaf::util::logging`

7.852 src/main/decaf/util/logging/LogManager.h File Reference

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::logging::LogManager**

*There is a single global **LogManager** (p. 2254) object that is used to maintain a set of shared state about Loggers and log services.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::io**
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.853 src/main/decaf/util/logging/LogRecord.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Level.h>
#include <decaf/util/Config.h>
#include <memory>
#include <string>
```

Data Structures

- class **decaf::util::logging::LogRecord**

***LogRecord** (p. 2261) objects are used to pass logging requests between the logging framework and individual log Handlers.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.854 src/main/decaf/util/logging/LogWriter.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::logging::LogWriter**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.855 src/main/decaf/util/logging/MarkBlockLogger.h File Reference

```
#include <decaf/util/logging/Logger.h>
```

Data Structures

- class **decaf::util::logging::MarkBlockLogger**

Defines a class that can be used to mark the entry and exit from scoped blocks.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.856 src/main/decaf/util/logging/PropertiesChangeListener.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::PropertiesChangeListener**

*Defines the interface that classes can use to listen for change events on **Properties** (p. 2927).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.857 src/main/decaf/util/logging/SimpleFormatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Formatter.h>
#include <string>
```

Data Structures

- class **decaf::util::logging::SimpleFormatter**

*Print a brief summary of the **LogRecord** (p. 2261) in a human readable format.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.858 src/main/decaf/util/logging/SimpleLogger.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class `decaf::util::logging::SimpleLogger`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::util`
- namespace `decaf::util::logging`

7.859 src/main/decaf/util/logging/StreamHandler.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/Config.h>
```

Data Structures

- class `decaf::util::logging::StreamHandler`
*Stream based logging **Handler** (p. 1852).*

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::io`
- namespace `decaf::util`
- namespace `decaf::util::logging`

7.860 src/main/decaf/util/logging/XMLFormatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Formatter.h>
#include <decaf/util/logging/LogRecord.h>
```

Data Structures

- class **decaf::util::logging::XMLFormatter**
*Format a **LogRecord** (p. 2261) into a standard XML format.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.861 src/main/decaf/util/Map.h File Reference

```
#include <functional>
#include <vector>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::Map< K, V, COMPARATOR >**
***Map** (p. 2305) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **decaf::util::Map< K, V, COMPARATOR >::Entry**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.862 src/main/decaf/util/PriorityQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Collection.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Comparator.h>
#include <decaf/util/comparators/Less.h>
#include <decaf/lang/Math.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <memory>
```

Data Structures

- class **decaf::util::PriorityQueue**< E >
An unbounded priority queue based on a binary heap algorithm.
- class **decaf::util::PriorityQueue**< E >::PriorityQueueIterator
- class **decaf::util::PriorityQueue**< E >::ConstPriorityQueueIterator

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.863 src/main/decaf/util/Properties.h File Reference

```
#include <vector>
#include <string>
#include <decaf/util/Config.h>
#include <decaf/util/StlMap.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::util::Properties**

Java-like properties class for mapping string names to string values.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**
- namespace **decaf::util**

7.864 src/main/decaf/util/Random.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Config.h>
#include <vector>
#include <cmath>
```

Data Structures

- class **decaf::util::Random**

***Random** (p. 2953) Value Generator which is used to generate a stream of pseudorandom numbers.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.865 src/main/decaf/util/Set.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
```

Data Structures

- class **decaf::util::Set< E >**
A collection that contains no duplicate elements.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.866 src/main/decaf/util/StlList.h File Reference

```
#include <list>
#include <algorithm>
#include <memory>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/ListIterator.h>
#include <decaf/util/List.h>
```

Data Structures

- class **decaf::util::StlList< E >**
List (p. 2190) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.
- class **decaf::util::StlList< E >::StlListIterator**
- class **decaf::util::StlList< E >::ConstStlListIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.867 src/main/decaf/util/StlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

Data Structures

- class **decaf::util::StlMap**< K, V, COMPARATOR >
Map (p. 2305) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.868 src/main/decaf/util/StlQueue.h File Reference

```
#include <list>
#include <vector>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::StlQueue**< T >
The Queue (p. 2948) class accepts messages with an `psuh(m)` command where *m* is the message to be queued.
- class **decaf::util::StlQueue**< T >::QueueIterator

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.869 src/main/decaf/util/StlSet.h File Reference

```
#include <set>
#include <vector>
#include <memory>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractSet.h>
```

Data Structures

- class **decaf::util::StlSet< E >**
Set (p. 3220) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.
- class **decaf::util::StlSet< E >::SetIterator**
- class **decaf::util::StlSet< E >::ConstSetIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.870 src/main/decaf/util/StringTokenizer.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::util::StringTokenizer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.871 src/main/decaf/util/Timer.h File Reference

```
#include <memory>
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::Timer**
A facility for threads to schedule tasks for future execution in a background thread.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.872 src/main/decaf/util/TimerTask.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::TimerTask**
*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3543).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::util**

7.873 src/main/decaf/util/UUID.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <apr_uuid.h>
#include <string>
```

Data Structures

- class **decaf::util::UUID**

*A class that represents an immutable universally unique identifier (**UUID** (p. 3706)).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.874 src/main/decaf/util/zip/Adler32.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
```

Data Structures

- class **decaf::util::zip::Adler32**

*Clas that can be used to compute an Adler-32 **Checksum** (p. 1059) for a data stream.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.875 src/main/decaf/util/zip/CheckedInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
#include <decaf/io/FilterInputStream.h>
```

Data Structures

- class **decaf::util::zip::CheckedInputStream**

*An implementation of a `FilterInputStream` that will maintain a **Checksum** (p. 1059) of the bytes read, the **Checksum** (p. 1059) can then be used to verify the integrity of the input stream.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.876 src/main/decaf/util/zip/CheckedOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
#include <decaf/io/FilterOutputStream.h>
```

Data Structures

- class **decaf::util::zip::CheckedOutputStream**

*An implementation of a `FilterOutputStream` that will maintain a **Checksum** (p. 1059) of the bytes written, the **Checksum** (p. 1059) can then be used to verify the integrity of the output stream.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.877 src/main/decaf/util/zip/Checksum.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <vector>
```

Data Structures

- class **decaf::util::zip::Checksum**

*An interface used to represent **Checksum** (p. 1059) values in the Zip package.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.878 src/main/decaf/util/zip/CRC32.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
```

Data Structures

- class **decaf::util::zip::CRC32**

Class that can be used to compute a CRC-32 checksum for a data stream.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.879 src/main/decaf/util/zip/DataFormatException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::zip::DataFormatException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.880 src/main/decaf/util/zip/Deflater.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <vector>
```

Data Structures

- class **decaf::util::zip::Deflater**

This class compresses data using the DEFLATE algorithm (see `specification`).

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.881 src/main/decaf/util/zip/DeflaterOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/zip/Deflater.h>
#include <vector>
```

Data Structures

- class **decaf::util::zip::DeflaterOutputStream**

Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.882 src/main/decaf/util/zip/Inflater.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/zip/DataFormatException.h>
#include <vector>
```

Data Structures

- class **decaf::util::zip::Inflater**

This class uncompresses data that was compressed using the DEFLATE algorithm (see specification).

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.883 src/main/decaf/util/zip/InflaterInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/zip/Inflater.h>
#include <vector>
```

Data Structures

- class **decaf::util::zip::InflaterInputStream**

A FilterInputStream that decompresses data read from the wrapped InputStream instance.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.884 src/main/decaf/util/zip/ZipException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::util::zip::ZipException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::zip**

Index

- ~AbstractCollection
 - decaf::util::AbstractCollection, 146
- ~AbstractList
 - decaf::util::AbstractList, 157
- ~AbstractMap
 - decaf::util::AbstractMap, 158
- ~AbstractQueue
 - decaf::util::AbstractQueue, 159
- ~AbstractSequentialList
 - decaf::util::AbstractSequentialList, 162
- ~AbstractSet
 - decaf::util::AbstractSet, 163
- ~AbstractTransportFactory
 - activemq::transport::AbstractTransportFactory, 165
- ~ActiveMQAckHandler
 - activemq::core::ActiveMQAckHandler, 166
- ~ActiveMQBlobMessage
 - activemq::commands::ActiveMQBlobMessage, 167
- ~ActiveMQBlobMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 176
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 184
 - activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller, 172
 - activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 180
 - activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, 188
 - activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller, 192
- ~ActiveMQBytesMessage
 - activemq::commands::ActiveMQBytesMessage, 198
- ~ActiveMQBytesMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 215
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 231
 - activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 211
 - activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 219
- activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 223
- activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller, 227
- ~ActiveMQCPP
 - activemq::library::ActiveMQCPP, 278
- ~ActiveMQConnection
 - activemq::core::ActiveMQConnection, 238
- ~ActiveMQConnectionFactory
 - activemq::core::ActiveMQConnectionFactory, 254
- ~ActiveMQConnectionMetaData
 - activemq::core::ActiveMQConnectionMetaData, 263
- ~ActiveMQConsumer
 - activemq::core::ActiveMQConsumer, 271
- ~ActiveMQDestination
 - activemq::commands::ActiveMQDestination, 282
- ~ActiveMQDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 295
 - activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 301
 - activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 307
 - activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 311
 - activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 313
 - activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller, 315
- ~ActiveMQException
 - activemq::exceptions::ActiveMQException, 315
- ~ActiveMQMapMessage
 - activemq::commands::ActiveMQMapMessage, 335
- ~ActiveMQMapMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 334
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 346
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 330

- activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 458
- activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 442
- activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller, 450
- ~ActiveMQMessage
 - activemq::commands::ActiveMQMessage, 353
- ~ActiveMQMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 360
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 372
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 356
 - activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 364
 - activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 368
 - activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller, 376
- ~ActiveMQMessageTemplate
 - activemq::commands::ActiveMQMessageTemplate, 383
- ~ActiveMQObjectMessage
 - activemq::commands::ActiveMQObjectMessage, 397
- ~ActiveMQObjectMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 404
 - activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 416
 - activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 400
 - activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 408
 - activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 412
 - activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller, 420
- ~ActiveMQProducer
 - activemq::core::ActiveMQProducer, 425
- ~ActiveMQProperties
 - activemq::util::ActiveMQProperties, 432
- ~ActiveMQQueue
 - activemq::commands::ActiveMQQueue, 436
- ~ActiveMQQueueBrowser
 - activemq::core::ActiveMQQueueBrowser, 440
- ~ActiveMQQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 446
- activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, 458
- activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 442
- activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller, 450
- activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, 454
- activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller, 462
- ~ActiveMQSession
 - activemq::core::ActiveMQSession, 469
- ~ActiveMQSessionExecutor
 - activemq::core::ActiveMQSessionExecutor, 483
- ~ActiveMQStreamMessage
 - activemq::commands::ActiveMQStreamMessage, 489
- ~ActiveMQStreamMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 505
 - activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 517
 - activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 501
 - activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 509
 - activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 513
 - activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller, 521
- ~ActiveMQTempDestination
 - activemq::commands::ActiveMQTempDestination, 525
- ~ActiveMQTempDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 531
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 543
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 528
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 535
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 539
 - activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller, 546
- ~ActiveMQTempQueue
 - activemq::commands::ActiveMQTempQueue, 550
- ~ActiveMQTempQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 558

- activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 570
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 554
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 562
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 566
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller, 574
- ~ActiveMQTempTopic
 - activemq::commands::ActiveMQTempTopic, 578
- ~ActiveMQTempTopicMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 590
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 598
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 582
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 586
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 594
 - activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller, 602
- ~ActiveMQTextMessage
 - activemq::commands::ActiveMQTextMessage, 606
- ~ActiveMQTextMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 618
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 630
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 610
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 614
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 622
 - activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller, 626
- ~ActiveMQTopic
 - activemq::commands::ActiveMQTopic, 634
- ~ActiveMQTopicMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 646
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 658
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 638
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 642
- ActiveMQTempQueueMarshaller, 650
- ActiveMQTempQueueMarshaller, 654
- ActiveMQTempQueueMarshaller, 662
- ActiveMQTempQueueMarshaller, 664
- ~Appendable
 - decaf::lang::Appendable, 667
- ~AprPool
 - decaf::internal::AprPool, 669
- ~ArrayPointer
 - decaf::lang::ArrayPointer, 673
- ~AtomicBoolean
 - decaf::util::concurrent::atomic::AtomicBoolean, 678
- ~AtomicInteger
 - decaf::util::concurrent::atomic::AtomicInteger, 681
- ~AtomicReference
 - decaf::util::concurrent::atomic::AtomicReference, 686
- ~BackupTransport
 - activemq::transport::failover::BackupTransport, 690
- ~BackupTransportPool
 - activemq::transport::failover::BackupTransportPool, 693
- ~BaseCommand
 - activemq::commands::BaseCommand, 696
- ~BaseCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 710
 - activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 730
 - activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 702
 - activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 709
 - activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 722
 - activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller, 729
- ~BaseMQStreamMarshaller
 - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 748
- ~BaseDataStructure
 - activemq::wireformat::openwire::marshal::BaseDataStructure, 765

- ~BindException
 - decaf::net::BindException, 770
- ~BlockingByteArrayInputStream
 - decaf::io::BlockingByteArrayInputStream, 772
- ~BlockingQueue
 - decaf::util::concurrent::BlockingQueue, 777
- ~Boolean
 - decaf::lang::Boolean, 782
- ~BooleanExpression
 - activemq::commands::BooleanExpression, 786
- ~BooleanStream
 - activemq::wireformat::openwire::utils::BooleanStream, 789
- ~BrokenBarrierException
 - decaf::util::concurrent::BrokenBarrierException, 792
- ~BrokerError
 - activemq::commands::BrokerError, 794
- ~BrokerException
 - activemq::exceptions::BrokerException, 797
- ~BrokerId
 - activemq::commands::BrokerId, 799
- ~BrokerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 810
 - activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 822
 - activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 802
 - activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 806
 - activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 814
 - activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller, 818
- ~BrokerInfo
 - activemq::commands::BrokerInfo, 826
- ~BrokerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 841
 - activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 853
 - activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 833
 - activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 837
 - activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 845
 - activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller, 849
- ~Buffer
 - decaf::nio::Buffer, 858
- ~BufferFactory
 - decaf::internal::nio::BufferFactory, 871
- ~BufferOverflowException
 - decaf::nio::BufferOverflowException, 882
- ~BufferUnderflowException
 - decaf::nio::BufferUnderflowException, 884
- ~BufferedInputStream
 - decaf::io::BufferedInputStream, 864
- ~BufferedOutputStream
 - decaf::io::BufferedOutputStream, 868
- ~Byte
 - decaf::lang::Byte, 887
- ~ByteArrayAdapter
 - decaf::internal::util::ByteArrayAdapter, 900
- ~ByteArrayBuffer
 - decaf::internal::nio::ByteArrayBuffer, 927
- ~ByteArrayInputStream
 - decaf::io::ByteArrayInputStream, 947
- ~ByteArrayOutputStream
 - decaf::io::ByteArrayOutputStream, 952
- ~ByteBuffer
 - decaf::nio::ByteBuffer, 959
- ~BytesMessage
 - cms::BytesMessage, 982
- ~CMSEncryptionException
 - cms::CMSEncryptionException, 1076
- ~CMSEncryptionSupport
 - activemq::util::CMSEncryptionSupport, 1078
- ~CMSProperties
 - cms::CMSProperties, 1080
- ~CMSSecurityException
 - cms::CMSSecurityException, 1083
- ~CRC32
 - decaf::util::CRC32, 1421
- ~CachedConsumer
 - activemq::cmsutil::CachedConsumer, 994
- ~CachedProducer
 - activemq::cmsutil::CachedProducer, 998
- ~Callable
 - decaf::util::concurrent::Callable, 1004
- ~CancellationException
 - decaf::util::concurrent::CancellationException, 1006
- ~Certificate
 - decaf::security::Certificate, 1008
- ~CertificateEncodingException
 - decaf::security::CertificateEncodingException, 1011
- ~CertificateException
 - decaf::security::CertificateException, 1013

- ~CertificateExpiredException
 - decaf::security::cert::CertificateExpiredException, 1015
- ~CertificateNotYetValidException
 - decaf::security::cert::CertificateNotYetValidException, 1017
- ~CertificateParsingException
 - decaf::security::cert::CertificateParsingException, 1019
- ~CharArrayBuffer
 - decaf::internal::nio::CharArrayBuffer, 1032
- ~CharBuffer
 - decaf::nio::CharBuffer, 1041
- ~CharSequence
 - decaf::lang::CharSequence, 1054
- ~CheckedInputStream
 - decaf::util::zip::CheckedInputStream, 1056
- ~CheckedOutputStream
 - decaf::util::zip::CheckedOutputStream, 1059
- ~Checksum
 - decaf::util::zip::Checksum, 1060
- ~ClassCastException
 - decaf::lang::exceptions::ClassCastException, 1064
- ~CloseTransportsTask
 - activemq::transport::failover::CloseTransportsTask, 1067
- ~Closeable
 - cms::Closeable, 1065
 - decaf::io::Closeable, 1066
- ~CmsAccessor
 - activemq::cmsutil::CmsAccessor, 1069
- ~CmsDestinationAccessor
 - activemq::cmsutil::CmsDestinationAccessor, 1073
- ~CmsTemplate
 - activemq::cmsutil::CmsTemplate, 1087
- ~Collection
 - decaf::util::Collection, 1099
- ~Command
 - activemq::commands::Command, 1108
- ~CommandVisitor
 - activemq::state::CommandVisitor, 1115
- ~CommandVisitorAdapter
 - activemq::state::CommandVisitorAdapter, 1123
- ~Comparable
 - decaf::lang::Comparable, 1126
- ~Comparator
 - decaf::util::Comparator, 1128
- ~CompositeData
 - activemq::util::CompositeData, 1131
- ~CompositeTask
 - activemq::threads::CompositeTask, 1132
 - activemq::threads::CompositeTaskRunner, 1134
 - activemq::transport::CompositeTransport, 1136
- ~ConcurrentMap
 - decaf::util::concurrent::ConcurrentMap, 1137
- ~ConcurrentStlMap
 - decaf::util::concurrent::ConcurrentStlMap, 1145
- ~Condition
 - decaf::util::concurrent::locks::Condition, 1158
- ~ConditionHandle
 - decaf::util::concurrent::ConditionHandle, 1163
- ~ConnectException
 - decaf::net::ConnectException, 1167
- ~Connection
 - cms::Connection, 1169
- ~ConnectionControl
 - activemq::commands::ConnectionControl, 1173
- ~ConnectionControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1186
 - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1198
 - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1178
 - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1182
 - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1190
 - activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 1194
- ~ConnectionError
 - activemq::commands::ConnectionError, 1202
- ~ConnectionErrorMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1217
 - activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1205
 - activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1209
 - activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1213
 - activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1221

- activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1316
- activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1316
- ~ConnectionFactory
 - cms::ConnectionFactory, 1229
- ~ConnectionId
 - activemq::commands::ConnectionId, 1322
- ~ConnectionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1247
 - activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1235
 - activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1239
 - activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1243
 - activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1251
 - activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 1255
- ~ConnectionInfo
 - activemq::commands::ConnectionInfo, 1259
- ~ConnectionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1277
 - activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1265
 - activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1269
 - activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1273
 - activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1281
 - activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1285
- ~ConnectionMetaData
 - cms::ConnectionMetaData, 1288
- ~ConnectionState
 - activemq::state::ConnectionState, 1292
- ~ConnectionStateTracker
 - activemq::state::ConnectionStateTracker, 1296
- ~ConsoleHandler
 - decaf::util::logging::ConsoleHandler, 1301
- ~ConsumerControl
 - activemq::commands::ConsumerControl, 1303
- ~ConsumerControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1320
 - activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1308
 - activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1312
- activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller, 1316
- activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1324
- activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller, 1328
- ~ConsumerId
 - activemq::commands::ConsumerId, 1332
- ~ConsumerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1347
 - activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1335
 - activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1339
 - activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1343
 - activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1351
 - activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 1355
- ~ConsumerInfo
 - activemq::commands::ConsumerInfo, 1360
- ~ConsumerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1379
 - activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1367
 - activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1371
 - activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1375
 - activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1383
 - activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1387
- ~ConsumerState
 - activemq::state::ConsumerState, 1390
- ~ControlCommand
 - activemq::commands::ControlCommand, 1391
- ~ControlCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1406
 - activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1394
 - activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1398
 - activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1402
 - activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1410
 - activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller, 1414

- ~CountDownLatch
 - decaf::util::concurrent::CountDownLatch, 1418
- ~DataArrayResponse
 - activemq::commands::DataArrayResponse, 1424
- ~DataArrayResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1439
 - activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1427
 - activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1431
 - activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1435
 - activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1443
 - activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller, 1447
- ~DataFormatException
 - decaf::util::zip::DataFormatException, 1451
- ~DataInput
 - decaf::io::DataInput, 1454
- ~DataInputStream
 - decaf::io::DataInputStream, 1462
- ~DataOutput
 - decaf::io::DataOutput, 1470
- ~DataOutputStream
 - decaf::io::DataOutputStream, 1475
- ~DataResponse
 - activemq::commands::DataResponse, 1478
- ~DataResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1501
 - activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1489
 - activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1493
 - activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1497
 - activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1481
 - activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 1485
- ~DataStreamMarshaller
 - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1505
- ~DataStructure
 - activemq::commands::DataStructure, 1554
- ~Date
 - decaf::util::Date, 1560
- ~DecafRuntime
 - decaf::internal::DecafRuntime, 1563
- ~DedicatedTaskRunner
 - activemq::threads::DedicatedTaskRunner, 1565
- ~DefaultPrefetchPolicy
 - activemq::core::policies::DefaultPrefetchPolicy, 1567
- ~DefaultRedeliveryPolicy
 - activemq::core::policies::DefaultRedeliveryPolicy, 1570
- ~DefaultSSLContextMarshaller
 - decaf::internal::net::ssl::DefaultSSLContext, 1582
- ~DefaultSSLServerSocketFactory
 - activemq::core::policies::DefaultSSLServerSocketFactory, 1584
- ~DefaultSSLSocketFactory
 - decaf::internal::net::ssl::DefaultSSLSocketFactory, 1586
- ~DefaultServerSocketFactory
 - decaf::internal::net::DefaultServerSocketFactory, 1575
- ~DefaultSocketFactory
 - decaf::internal::net::DefaultSocketFactory, 1579
- ~DefaultTransportListener
 - activemq::transport::DefaultTransportListener, 1594
- ~Deflater
 - decaf::util::zip::Deflater, 1597
- ~DeflaterOutputStream
 - decaf::util::zip::DeflaterOutputStream, 1607
- ~Delayed
 - activemq::core::Delayed, 1609
- ~DeliveryMode
 - activemq::core::DeliveryMode, 1610
- ~Destination
 - activemq::core::Destination, 1612
- ~DestinationInfo
 - activemq::core::DestinationInfo, 1615
- ~DestinationInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1634
 - activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1619
 - activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1623
 - activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1627
 - activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1639
 - activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1635

- ~DestinationResolver
 - activemq::cmsutil::DestinationResolver, 1642
- ~DiscoveryEvent
 - activemq::commands::DiscoveryEvent, 1645
- ~DiscoveryEventMarshaller
 - activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1664
 - activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1652
 - activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1656
 - activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1660
 - activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1668
 - activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1648
- ~Dispatcher
 - activemq::core::Dispatcher, 1672
- ~Double
 - decaf::lang::Double, 1674
- ~DoubleArrayBuffer
 - decaf::internal::nio::DoubleArrayBuffer, 1688
- ~DoubleBuffer
 - decaf::nio::DoubleBuffer, 1695
- ~DynamicDestinationResolver
 - activemq::cmsutil::DynamicDestinationResolver, 1705
- ~EOFException
 - decaf::io::EOFException, 1709
- ~Entry
 - decaf::util::Map::Entry, 1707
- ~ErrorManager
 - decaf::util::logging::ErrorManager, 1711
- ~Exception
 - decaf::lang::Exception, 1715
- ~ExceptionListener
 - cms::ExceptionListener, 1719
- ~ExceptionResponse
 - activemq::commands::ExceptionResponse, 1721
- ~ExceptionResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1744
 - activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1728
 - activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1732
 - activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1740
- activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1736
- activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 1724
- ~ExecutionException
 - decaf::util::concurrent::ExecutionException, 1748
- ~Executor
 - decaf::util::concurrent::Executor, 1750
- ~ExecutorService
 - decaf::util::concurrent::ExecutorService, 1752
- ~FailoverTransport
 - activemq::transport::failover::FailoverTransport, 1755
- ~FailoverTransportFactory
 - activemq::transport::failover::FailoverTransportFactory, 1764
- ~FailoverTransportListener
 - activemq::transport::failover::FailoverTransportListener, 1767
- ~FileDescriptor
 - decaf::io::FileDescriptor, 1769
- ~Filter
 - decaf::util::logging::Filter, 1770
- ~FilterInputStream
 - decaf::io::FilterInputStream, 1773
- ~FilterOutputStream
 - decaf::io::FilterOutputStream, 1778
- ~Float
 - decaf::lang::Float, 1783
- ~FloatArrayBuffer
 - decaf::internal::nio::FloatArrayBuffer, 1795
- ~FloatBuffer
 - decaf::nio::FloatBuffer, 1803
- ~FlushCommand
 - activemq::commands::FlushCommand, 1813
- ~FlushCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1832
 - activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 1820
 - activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 1824
 - activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 1828
 - activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 1836
 - activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, 1846
- ~Flushable
 - decaf::util::concurrent::Flushable, 1841
- ~Formatter

- decaf::util::logging::Formatter, 1839
- ~Future
 - decaf::util::concurrent::Future, 1841
- ~FutureResponse
 - activemq::transport::correlator::FutureResponse, 1844
- ~GeneralSecurityException
 - decaf::security::GeneralSecurityException, 1847
- ~GenericResource
 - decaf::internal::util::GenericResource, 1848
- ~Handler
 - decaf::util::logging::Handler, 1853
- ~HexStringParser
 - decaf::internal::util::HexStringParser, 1856
- ~HexTable
 - activemq::wireformat::openwire::utils::HexTable, 1858
- ~HttpRequestException
 - decaf::net::HttpRequestException, 1860
- ~IOException
 - decaf::io::IOException, 2005
- ~IOTransport
 - activemq::transport::IOTransport, 2007
- ~IdGenerator
 - activemq::util::IdGenerator, 1862
- ~IllegalArgumentException
 - decaf::lang::exceptions::IllegalArgumentException, 1865
- ~IllegalMonitorStateException
 - decaf::lang::exceptions::IllegalMonitorStateException, 1867
- ~IllegalStateException
 - cms::IllegalStateException, 1869
 - decaf::lang::exceptions::IllegalStateException, 1871
- ~IllegalThreadStateException
 - decaf::lang::exceptions::IllegalThreadStateException, 1873
- ~InactivityMonitor
 - activemq::transport::inactivity::InactivityMonitor, 1875
- ~IndexOutOfBoundsException
 - decaf::lang::exceptions::IndexOutOfBoundsException, 1879
- ~Inet4Address
 - decaf::net::Inet4Address, 1881
- ~Inet6Address
 - decaf::net::Inet6Address, 1884
- ~InetAddress
 - decaf::net::InetAddress, 1887
- ~InetSocketAddress
 - decaf::net::InetSocketAddress, 1892
- ~Inflater
 - decaf::util::zip::Inflater, 1896
- ~InflaterInputStream
 - decaf::util::zip::InflaterInputStream, 1905
- ~InputStream
 - decaf::io::InputStream, 1911
- ~InputStreamReader
 - decaf::io::InputStreamReader, 1920
- ~IntArrayBuffer
 - decaf::internal::nio::IntArrayBuffer, 1926
- ~IntBuffer
 - decaf::nio::IntBuffer, 1933
- ~Integer
 - decaf::lang::Integer, 1945
- ~IntegerResponse
 - activemq::commands::IntegerResponse, 1957
- ~IntegerResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 1976
 - activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 1964
 - activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 1968
 - activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 1972
 - activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 1980
 - activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 1960
- ~InternalCommandListener
 - activemq::transport::mock::InternalCommandListener, 1987
- ~InterruptedException
 - decaf::lang::exceptions::InterruptedException, 1989
- ~InterruptedIOException
 - decaf::io::InterruptedIOException, 1992
- ~InvalidClientIdException
 - cms::InvalidClientIdException, 1993
- ~InvalidDestinationException
 - cms::InvalidDestinationException, 1994
- ~InvalidKeyException
 - decaf::security::InvalidKeyException, 1996
- ~InvalidMarkException
 - decaf::nio::InvalidMarkException, 1999
- ~InvalidSelectorException
 - cms::InvalidSelectorException, 2000
- ~InvalidStateException
 - decaf::lang::exceptions::InvalidStateException, 2002
- ~Iterable
 - decaf::lang::Iterable, 2012
- ~Iterator
 - decaf::util::Iterator, 2013

- ~JournalQueueAck
 - activemq::commands::JournalQueueAck, 2015
- ~JournalQueueAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2038
 - activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 2022
 - activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 2030
 - activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 2034
 - activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 2026
 - activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, 2018
- ~JournalTopicAck
 - activemq::commands::JournalTopicAck, 2042
- ~JournalTopicAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2067
 - activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 2051
 - activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 2055
 - activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 2063
 - activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 2047
 - activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, 2059
- ~JournalTrace
 - activemq::commands::JournalTrace, 2070
- ~JournalTraceMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2089
 - activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 2073
 - activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 2077
 - activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 2085
 - activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 2093
 - activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller, 2081
- ~JournalTransaction
 - activemq::commands::JournalTransaction, 2096
- ~JournalTransactionMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2120
- activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2104
- activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 2108
- activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 2116
- activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 2112
- activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, 2100
- ~KeepAliveInfo
 - activemq::commands::KeepAliveInfo, 2123
- ~KeepAliveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2146
 - activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 2130
 - activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 2134
 - activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 2138
 - activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 2142
 - activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2126
- ~Key
 - decaf::security::Key, 2150
- ~KeyException
 - decaf::security::KeyException, 2153
- ~KeyManagementException
 - decaf::security::KeyManagementException, 2155
- ~LastPartialCommand
 - activemq::commands::LastPartialCommand, 2157
- ~LastPartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2179
 - activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 2167
 - activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 2163
 - activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 2175
 - activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 2171
 - activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, 2159
- ~Less
 - decaf::util::comparators::Less, 2183
- ~Level
 - decaf::util::logging::Level, 2187
- ~List
 - decaf::util::List, 2192

- ~ListIterator
 - decaf::util::ListIterator, 2198
- ~LocalTransactionId
 - activemq::commands::LocalTransactionId, 2202
- ~LocalTransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2225
 - activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2209
 - activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2213
 - activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 2221
 - activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2217
 - activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2205
- ~Lock
 - decaf::util::concurrent::Lock, 2229
 - decaf::util::concurrent::locks::Lock, 2231
- ~LockSupport
 - decaf::util::concurrent::locks::LockSupport, 2236
- ~LogManager
 - decaf::util::logging::LogManager, 2257
- ~LogRecord
 - decaf::util::logging::LogRecord, 2262
- ~LogWriter
 - decaf::util::logging::LogWriter, 2266
- ~Logger
 - decaf::util::logging::Logger, 2241
- ~LoggerHierarchy
 - decaf::util::logging::LoggerHierarchy, 2249
- ~LoggingInputStream
 - activemq::io::LoggingInputStream, 2250
- ~LoggingOutputStream
 - activemq::io::LoggingOutputStream, 2251
- ~LoggingTransport
 - activemq::transport::logging::LoggingTransport, 2253
- ~Long
 - decaf::lang::Long, 2270
- ~LongArrayBuffer
 - decaf::internal::nio::LongArrayBuffer, 2286
- ~LongBuffer
 - decaf::nio::LongBuffer, 2294
- ~LongSequenceGenerator
 - activemq::util::LongSequenceGenerator, 2302
- ~MalformedURLException
 - decaf::net::MalformedURLException, 2305
- ~Map
 - decaf::util::Map, 2307
- ~MapMessage
 - cms::MapMessage, 2320
- ~MarkBlockLogger
 - decaf::util::logging::MarkBlockLogger, 2328
- ~MarshalAware
 - activemq::wireformat::openwire::marshal::v1::MarshalAware, 2329
- ~MarshallerFactory
 - activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 2334
 - activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2335
 - activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 2332
 - activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 2333
 - activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2333
 - activemq::wireformat::openwire::marshal::v6::MarshallerFactory, 2331
- ~MarshallingSupport
 - activemq::util::MarshallingSupport, 2336
- ~Math
 - decaf::lang::Math, 2341
- ~MemoryUsage
 - activemq::util::MemoryUsage, 2356
- ~Message
 - activemq::commands::Message, 2362
 - cms::Message, 2379
- ~MessageAck
 - activemq::commands::MessageAck, 2395
- ~MessageAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2416
 - activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2404
 - activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2408
 - activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2412
 - activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2420
 - activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2400
- ~MessageConsumer
 - cms::MessageConsumer, 2424
- ~MessageCreator
 - activemq::cmsutil::MessageCreator, 2426
- ~MessageDispatch
 - activemq::commands::MessageDispatch, 2428
- ~MessageDispatchChannel
 - activemq::core::MessageDispatchChannel, 2433

- ~MessageDispatchMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2455
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2439
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2443
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2451
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2447
 - activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2459
- ~MessageDispatchNotification
 - activemq::commands::MessageDispatchNotification, 2463
- ~MessageDispatchNotificationMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2484
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2472
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2476
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2480
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2488
 - activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2468
- ~MessageEOFException
 - cms::MessageEOFException, 2493
- ~MessageEnumeration
 - cms::MessageEnumeration, 2491
- ~MessageFormatException
 - cms::MessageFormatException, 2494
- ~MessageId
 - activemq::commands::MessageId, 2496
- ~MessageIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2520
 - activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2500
 - activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2512
 - activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2504
 - activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2508
 - activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 2516
- ~MessageListener
 - cms::MessageListener, 2523
- ~MessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2541
 - activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2533
 - activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2529
 - activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2537
 - activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2524
 - activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2545
- ~MessageNotReadableException
 - cms::MessageNotReadableException, 2549
- ~MessageNotWritableException
 - cms::MessageNotWritableException, 2550
- ~MessageProducer
 - cms::MessageProducer, 2552
- ~MessagePropertyInterceptor
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2559
- ~MessagePull
 - activemq::commands::MessagePull, 2565
- ~MessagePullMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2585
 - activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2569
 - activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2577
 - activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2581
 - activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2573
 - activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2589
- ~MockTransport
 - activemq::transport::mock::MockTransport, 2594
- ~MockTransportFactory
 - activemq::transport::mock::MockTransportFactory, 2603
- ~Mutex
 - activemq::util::concurrent::Mutex, 2605
- ~MutexHandle
 - activemq::util::concurrent::MutexHandle, 2609
- ~Network
 - activemq::net::Network, 2612
- ~NetworkBridgeFilter
 - activemq::commands::NetworkBridgeFilter, 2615
- ~NetworkBridgeFilterMarshaller
 - activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2638

- activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2618
- activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2697
- activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2630
- decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2699
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2634
- ~OpenWireFormat, 2699
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2626
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2703
- activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2622
- activemq::wireformat::openwire::OpenWireFormatFactory, 2712
- ~NoRouteToHostException, 2712
- decaf::net::NoRouteToHostException, 2642
- ~OpenWireFormatNegotiator, 2714
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2714
- ~NoSuchAlgorithmException, 2714
- decaf::security::NoSuchAlgorithmException, 2645
- ~OpenWireResponseBuilder, 2718
- activemq::wireformat::openwire::OpenWireResponseBuilder, 2718
- ~NoSuchElementException, 2718
- decaf::lang::exceptions::NoSuchElementException, 2647
- ~OutputStream, 2720
- decaf::io::OutputStream, 2720
- ~NoSuchProviderException, 2720
- decaf::security::NoSuchProviderException, 2650
- ~OutputStreamWriter, 2727
- decaf::io::OutputStreamWriter, 2727
- ~NullPointerException, 2729
- decaf::lang::exceptions::NullPointerException, 2652
- ~PartialCommand, 2729
- activemq::commands::PartialCommand, 2729
- ~PartialCommandMarshaller, 2753
- activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2753
- ~Number, 2736
- activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2736
- ~NumberFormatException, 2745
- decaf::lang::exceptions::NumberFormatException, 2657
- activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2745
- ~ObjectMessage, 2749
- cms::ObjectMessage, 2658
- activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2749
- ~OpenSSLContextSpi, 2741
- decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2660
- activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2741
- ~OpenSSLParameters, 2732
- decaf::internal::net::ssl::openssl::OpenSSLParameters, 2662
- activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2732
- ~OpenSSLServerSocket, 2760
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2666
- ~Pointer, 2760
- decaf::lang::Pointer, 2760
- ~OpenSSLServerSocketFactory, 2778
- decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2671
- ~PooledSession, 2767
- activemq::cmsutil::PooledSession, 2767
- ~OpenSSLServerSocketFactory, 2778
- decaf::util::concurrent::PooledThread, 2778
- ~PooledThread, 2780
- ~PooledThreadListener, 2780
- decaf::util::concurrent::PooledThreadListener, 2780
- ~PortUnreachableException, 2783
- decaf::net::PortUnreachableException, 2689
- ~PrefetchPolicy, 2785
- activemq::core::PrefetchPolicy, 2785
- ~OpenSSLSocketException, 2790
- decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2692
- ~PrimitiveList, 2790
- activemq::util::PrimitiveList, 2790
- ~OpenSSLSocketFactory, 2790
- ~PrimitiveMap, 2790
- ~OpenSSLSocketInputStream, 2790

- activemq::util::PrimitiveMap, 2800
- ~PrimitiveTypesMarshaller
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2810
- ~PrimitiveValueConverter
 - activemq::util::PrimitiveValueConverter, 2817
- ~PrimitiveValueNode
 - activemq::util::PrimitiveValueNode, 2824
- ~Principal
 - decaf::security::Principal, 2831
- ~PriorityQueue
 - decaf::util::PriorityQueue, 2835
- ~ProducerAck
 - activemq::commands::ProducerAck, 2840
- ~ProducerAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 2864
 - activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 2844
 - activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 2852
 - activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 2848
 - activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 2856
 - activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 2860
- ~ProducerCallback
 - activemq::cmsutil::ProducerCallback, 2867
- ~ProducerExecutor
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2868
- ~ProducerId
 - activemq::commands::ProducerId, 2871
- ~ProducerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 2895
 - activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 2875
 - activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 2883
 - activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 2879
 - activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 2887
 - activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 2891
- ~ProducerInfo
 - activemq::commands::ProducerInfo, 2899
- ~ProducerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 2911
- activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 2907
- activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 2919
- activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 2903
- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2915
- activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 2923
- ~ProducerState
 - activemq::state::ProducerState, 2926
- ~Properties
 - decaf::util::Properties, 2929
- ~PropertiesChangeListener
 - decaf::util::logging::PropertiesChangeListener, 2937
- ~ProtocolException
 - decaf::net::ProtocolException, 2939
- ~PublicKey
 - decaf::security::PublicKey, 2940
- ~PushbackInputStream
 - decaf::io::PushbackInputStream, 2943
- ~Queue
 - cms::Queue, 2947
 - decaf::util::Queue, 2949
- ~QueueBrowser
 - cms::QueueBrowser, 2952
- ~ReadChecker
 - activemq::transport::inactivity::ReadChecker, 2960
- ~ReadOnlyBufferException
 - decaf::nio::ReadOnlyBufferException, 2968
- ~ReadWriteLock
 - decaf::util::concurrent::locks::ReadWriteLock, 2970
- ~Readable
 - decaf::lang::Readable, 2958
- ~Reader
 - decaf::io::Reader, 2962
- ~ReceiveExecutor
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2971
- ~RedeliveryPolicy
 - activemq::core::RedeliveryPolicy, 2974
- ~ReentrantLock
 - decaf::util::concurrent::locks::ReentrantLock, 2976
- ~RejectedExecutionException
 - decaf::util::concurrent::RejectedExecutionException, 2986
- ~RejectedExecutionHandler
 - decaf::util::concurrent::RejectedExecutionHandler, 2987

- ~RemoveInfo
 - activemq::commands::RemoveInfo, 2989
- ~RemoveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 3004
 - activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 2992
 - activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 3000
 - activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 3012
 - activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 3008
 - activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 2996
- ~RemoveSubscriptionInfo
 - activemq::commands::RemoveSubscriptionInfo, 3016
- ~RemoveSubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3020
 - activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3028
 - activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3024
 - activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3040
 - activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3036
 - activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3032
- ~ReplayCommand
 - activemq::commands::ReplayCommand, 3044
- ~ReplayCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 3051
 - activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 3055
 - activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 3059
 - activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 3047
 - activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 3067
 - activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3063
- ~ResolveProducerExecutor
 - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3071
- ~ResolveReceiveExecutor
 - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3072
- ~Resource
 - decaf::internal::util::Resource, 3073
- ~ResourceLifecycleManager
 - activemq::cmsutil::ResourceLifecycleManager, 3071
 - decaf::internal::util::ResourceLifecycleManager, 3071
- ~Response
 - activemq::commands::Response, 3077
- ~ResponseBuilder
 - activemq::transport::mock::ResponseBuilder, 3080
- ~ResponseCorrelator
 - activemq::transport::correlator::ResponseCorrelator, 3082
- ~ResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 3104
 - activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 3090
 - activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 3099
 - activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 3086
 - activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 3095
 - activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3108
- ~Runnable
 - decaf::lang::Runnable, 3112
- ~Runtime
 - decaf::lang::Runtime, 3113
- ~RuntimeException
 - decaf::lang::exceptions::RuntimeException, 3116
- ~SSLContext
 - ReplayCommandMarshaller::SSLContext, 3322
- ~SSLContextSpi
 - ReplayCommandMarshaller::SSLContextSpi, 3324
- ~SSLParameters
 - ReplayCommandMarshaller::SSLParameters, 3328
- ~SSLServerSocket
 - ReplayCommandMarshaller::SSLServerSocket, 3332
- ~SSLServerSocketFactory
 - ReplayCommandMarshaller::SSLServerSocketFactory, 3336
- ~SSLSocketFactory
 - decaf::net::ssl::SSLSocket, 3340
- ~SecureRandom
 - decaf::security::SecureRandom, 3119
- ~SecureRandomImpl
 - decaf::internal::security::SecureRandomImpl, 3122

- ~SecureRandomSpi
 - decaf::security::SecureRandomSpi, 3125
- ~Semaphore
 - decaf::util::concurrent::Semaphore, 3129
- ~SendExecutor
 - activemq::cmsutil::CmsTemplate::SendExecutor, 3136
- ~ServerSocket
 - decaf::net::ServerSocket, 3140
- ~ServerSocketFactory
 - decaf::net::ServerSocketFactory, 3146
- ~Session
 - cms::Session, 3152
- ~SessionCallback
 - activemq::cmsutil::SessionCallback, 3161
- ~SessionId
 - activemq::commands::SessionId, 3163
- ~SessionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 3186
 - activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 3166
 - activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 3182
 - activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 3170
 - activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 3178
 - activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3174
- ~SessionInfo
 - activemq::commands::SessionInfo, 3189
- ~SessionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 3201
 - activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 3209
 - activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 3205
 - activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 3213
 - activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 3197
 - activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3193
- ~SessionPool
 - activemq::cmsutil::SessionPool, 3216
- ~SessionState
 - activemq::state::SessionState, 3219
- ~Set
 - decaf::util::Set, 3220
- ~Short
 - decaf::lang::Short, 3223
- ~ShortArrayBuffer
 - decaf::internal::nio::ShortArrayBuffer, 3234
- ~ShortBuffer
 - decaf::nio::ShortBuffer, 3241
- ~ShutdownInfo
 - activemq::commands::ShutdownInfo, 3250
- ~ShutdownInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 3261
 - activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 3257
 - activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 3269
 - activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 3273
 - activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 3265
 - activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3253
- ~SignatureException
 - decaf::security::SignatureException, 3278
- ~SimpleFormatter
 - decaf::util::logging::SimpleFormatter, 3279
- ~SimpleLogger
 - decaf::util::logging::SimpleLogger, 3280
- ~Socket
 - decaf::net::Socket, 3287
- ~SocketAddress
 - decaf::net::SocketAddress, 3298
- ~SocketException
 - decaf::net::SocketException, 3300
- ~SocketFactory
 - decaf::net::SocketFactory, 3302
- ~SocketFileDescriptor
 - decaf::internal::net::SocketFileDescriptor, 3305
- ~SocketImpl
 - decaf::net::SocketImpl, 3308
- ~SocketImplFactory
 - decaf::net::SocketImplFactory, 3314
- ~SocketOptions
 - decaf::net::SocketOptions, 3316
- ~SocketTimeoutException
 - decaf::net::SocketTimeoutException, 3320
- ~SslTransport
 - activemq::transport::tcp::SslTransport, 3348
- ~SslTransportFactory
 - activemq::transport::tcp::SslTransportFactory, 3350
- ~StandardErrorOutputStream
 - decaf::internal::io::StandardErrorOutputStream, 3352
- ~StandardInputStream

- decaf::internal::io::StandardInputStream, 3353
- ~StandardOutputStream
 - decaf::internal::io::StandardOutputStream, 3354
- ~Startable
 - cms::Startable, 3356
- ~StaticInitializer
 - activemq::core::ActiveMQConstants::StaticInitializer, 3357
- ~StlList
 - decaf::util::StlList, 3362
- ~StlMap
 - decaf::util::StlMap, 3374
- ~StlQueue
 - decaf::util::StlQueue, 3384
- ~StlSet
 - decaf::util::StlSet, 3392
- ~StompFrame
 - activemq::wireformat::stomp::StompFrame, 3399
- ~StompHelper
 - activemq::wireformat::stomp::StompHelper, 3404
- ~StompWireFormat
 - activemq::wireformat::stomp::StompWireFormat, 3408
- ~StompWireFormatFactory
 - activemq::wireformat::stomp::StompWireFormatFactory, 3411
- ~Stoppable
 - cms::Stoppable, 3412
- ~StreamHandler
 - decaf::util::logging::StreamHandler, 3413
- ~StreamMessage
 - cms::StreamMessage, 3418
- ~String
 - decaf::lang::String, 3429
- ~StringTokenizer
 - decaf::util::StringTokenizer, 3431
- ~SubscriptionInfo
 - activemq::commands::SubscriptionInfo, 3435
- ~SubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 3443
 - activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 3459
 - activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 3439
 - activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 3451
 - activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 3447
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfo, 3455
- ~Synchronizable
 - decaf::util::concurrent::Synchronizable, 3463
- ~SynchronizableImpl
 - decaf::internal::util::concurrent::SynchronizableImpl, 3474
- ~Synchronization
 - activemq::core::Synchronization, 3477
- ~SynchronousQueue
 - decaf::util::concurrent::SynchronousQueue, 3480
- ~System
 - decaf::lang::System, 3489
- ~Task
 - activemq::threads::Task, 3495
- ~TaskListener
 - decaf::util::concurrent::TaskListener, 3496
- ~TaskRunner
 - activemq::threads::TaskRunner, 3497
- ~TcpSocket
 - decaf::internal::net::tcp::TcpSocket, 3500
- ~TcpSocketInputStream
 - decaf::internal::net::tcp::TcpSocketInputStream, 3507
- ~TcpSocketOutputStream
 - decaf::internal::net::tcp::TcpSocketOutputStream, 3509
- ~TcpTransport
 - activemq::transport::tcp::TcpTransport, 3511
- ~TcpTransportFactory
 - activemq::transport::tcp::TcpTransportFactory, 3515
- ~TemporaryQueue
 - cms::TemporaryQueue, 3517
- ~TemporaryTopic
 - cms::TemporaryTopic, 3518
- ~TextMessage
 - cms::TextMessage, 3519
- ~Thread
 - decaf::lang::Thread, 3525
- ~ThreadFactory
 - decaf::util::concurrent::ThreadFactory, 3530
- ~ThreadGroup
 - decaf::lang::ThreadGroup, 3531
- ~ThreadPool
 - decaf::util::concurrent::ThreadPool, 3533
- ~Throwable
 - decaf::lang::Throwable, 3538
- ~TimeUnit
 - decaf::util::concurrent::TimeUnit, 3561

- ~TimeoutException
 - decaf::util::concurrent::TimeoutException, 3542
- ~Timer
 - decaf::util::Timer, 3545
- ~TimerTask
 - decaf::util::TimerTask, 3555
- ~TimerTaskHeap
 - decaf::internal::util::TimerTaskHeap, 3557
- ~Topic
 - cms::Topic, 3568
- ~Tracked
 - activemq::state::Tracked, 3569
- ~TransactionId
 - activemq::commands::TransactionId, 3571
- ~TransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3577
 - activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3581
 - activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3585
 - activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3588
 - activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3574
 - activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3592
- ~TransactionInfo
 - activemq::commands::TransactionInfo, 3596
- ~TransactionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3604
 - activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3620
 - activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3608
 - activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3616
 - activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3600
 - activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3612
- ~TransactionState
 - activemq::state::TransactionState, 3624
- ~TransferQueue
 - decaf::internal::util::concurrent::TransferQueue, 3626
- ~TransferStack
 - decaf::internal::util::concurrent::TransferStack, 3628
- ~Transport
 - activemq::transport::Transport, 3630
- ~TransportFactory
 - activemq::transport::TransportFactory, 3635
- ~TransportFilter
 - activemq::transport::TransportFilter, 3638
- ~TransportListener
 - activemq::transport::TransportListener, 3644
- ~TransportRegistry
 - activemq::transport::TransportRegistry, 3646
- ~URI
 - decaf::net::URI, 3664
- ~URIEncoderDecoder
 - decaf::internal::net::URIEncoderDecoder, 3673
- ~URIHelper
 - decaf::internal::net::URIHelper, 3676
- ~URIPool
 - activemq::transport::failover::URIPool, 3682
- ~URISyntaxException
 - decaf::net::URISyntaxException, 3689
- ~URIType
 - decaf::net::URIType, 3692
- ~URL
 - decaf::net::URL, 3699
- ~URLDecoder
 - decaf::net::URLDecoder, 3700
- ~URLEncoder
 - decaf::net::URLEncoder, 3701
- ~UTFDataFormatException
 - decaf::io::UTFDataFormatException, 3705
- ~UUID
 - decaf::util::UUID, 3708
- ~UncaughtExceptionHandler
 - decaf::internal::UncaughtExceptionHandler, 3649
- ~UnknownHostException
 - decaf::net::UnknownHostException, 3651
- ~UnknownServiceException
 - decaf::net::UnknownServiceException, 3653
- ~UnsupportedEncodingException
 - decaf::io::UnsupportedEncodingException, 3656
- ~UnsupportedOperationException
 - cms::UnsupportedOperationException, 3660
 - decaf::lang::exceptions::UnsupportedOperationException, 3659
- ~Usage
 - activemq::util::Usage, 3702
- ~WireFormat

- activemq::wireformat::WireFormat, 3714
- ~WireFormatFactory
 - activemq::wireformat::WireFormatFactory, 3717
- ~WireFormatInfo
 - activemq::commands::WireFormatInfo, 3720
- ~WireFormatInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 3745
 - activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 3737
 - activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 3749
 - activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 3741
 - activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3729
 - activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 3733
- ~WireFormatNegotiator
 - activemq::wireformat::WireFormatNegotiator, 3752
- ~WireFormatRegistry
 - activemq::wireformat::WireFormatRegistry, 3753
- ~WriteChecker
 - activemq::transport::inactivity::WriteChecker, 3755
- ~Writer
 - decaf::io::Writer, 3757
- ~X500Principal
 - decaf::security::auth::x500::X500Principal, 3762
- ~X509Certificate
 - decaf::security::cert::X509Certificate, 3764
- ~XATransactionId
 - activemq::commands::XATransactionId, 3766
- ~XATransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 3782
 - activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 3774
 - activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 3786
 - activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 3778
 - activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 3790
 - activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 3770
- ~XMLFormatter
 - decaf::util::logging::XMLFormatter, 3794
- ~ZipException
 - decaf::util::zip::ZipException, 3797
- _FALSE
 - decaf::lang::Boolean, 785
- _TRUE
 - decaf::lang::Boolean, 785
- _array
 - decaf::internal::nio::CharArrayBuffer, 1037
- _ByteBuffer
 - decaf::nio::Buffer, 861
- _deflate
 - deflate.h, 4202
- _length_code
 - trees.h, 4207
- _length_code
 - deflate.h, 4202
- _trees
 - trees.h, 4208
- _limit
 - decaf::nio::Buffer, 861
- _mark
 - decaf::nio::Buffer, 861
- _markSet
 - decaf::nio::Buffer, 861
- _position
 - decaf::nio::Buffer, 861
- _tr_tally_dist
 - deflate.h, 4201
- _tr_tally_lit
 - deflate.h, 4201
- ABORT
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- abs
 - decaf::lang::Math, 2341, 2342
- AbstractCollection
 - decaf::util::AbstractCollection, 146
- AbstractQueue
 - decaf::util::AbstractQueue, 159
- accept
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2666
 - decaf::internal::net::tcp::TcpSocket, 3501
 - decaf::net::ServerSocket, 3141
 - decaf::net::SocketImpl, 3308
- accepted
 - decaf::net::ServerSocket, 3141
- ACK
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- ACK_AUTO
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- ACK_CLIENT

- activemq::wireformat::stomp::StompCommandConstants, 3397
- ACK_INDIVIDUAL
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- ACK_TYPE_CONSUMED
 - activemq::core::ActiveMQConstants, 267
- ACK_TYPE_DELIVERED
 - activemq::core::ActiveMQConstants, 267
- ACK_TYPE_INDIVIDUAL
 - activemq::core::ActiveMQConstants, 267
- ACK_TYPE_POISON
 - activemq::core::ActiveMQConstants, 267
- ACK_TYPE_REDELIVERED
 - activemq::core::ActiveMQConstants, 267
- acknowledge
 - activemq::commands::ActiveMQMessageTemplate, 383
 - activemq::core::ActiveMQConsumer, 271
 - activemq::core::ActiveMQSession, 469
 - cms::Message, 2379
- acknowledgeMessage
 - activemq::core::ActiveMQAckHandler, 166
- AcknowledgeMode
 - cms::Session, 3151
- AckType
 - activemq::core::ActiveMQConstants, 267
- ackType
 - activemq::commands::MessageAck, 2399
- acquire
 - decaf::util::concurrent::Semaphore, 3129, 3130
- acquireUninterruptibly
 - decaf::util::concurrent::Semaphore, 3130, 3131
- action
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2869
- activemq, 83
- activemq/exceptions/ExceptionDefines.h
 - AMQ_CATCH_EXCEPTION_-CONVERT, 3855
 - AMQ_CATCH_NOTHROW, 3855
 - AMQ_CATCH_RETHROW, 3855
 - AMQ_CATCHALL_NOTHROW, 3856
 - AMQ_CATCHALL_THROW, 3856
- activemq/util/Config.h
 - AMQCPP_API, 3889
 - HAVE_PTHREAD_H, 3889
 - HAVE_UUID_T, 3889
 - HAVE_UUID_UUID_H, 3889
- activemq::cmsutil, 84
- activemq::cmsutil::CachedConsumer, 993
 - ~CachedConsumer, 994
- activemq::cmsutil::CachedConsumer, 994
 - close, 994
 - getMessageListener, 994
 - getMessageSelector, 994
 - operator=, 995
 - receive, 995
 - receiveNoWait, 995
 - setMessageListener, 996
- activemq::cmsutil::CachedProducer, 996
 - ~CachedProducer, 998
 - CachedProducer, 998
 - close, 998
 - getDeliveryMode, 998
 - getDisableMessageID, 998
 - getDisableMessageTimeStamp, 999
 - getPriority, 999
 - getTimeToLive, 999
 - operator=, 999
 - send, 1000, 1001
 - setDeliveryMode, 1001
 - setDisableMessageID, 1002
 - setDisableMessageTimeStamp, 1002
 - setPriority, 1002
 - setTimeToLive, 1003
- activemq::cmsutil::CmsAccessor, 1068
 - ~CmsAccessor, 1069
 - checkConnectionFactory, 1069
 - CmsAccessor, 1069
 - createConnection, 1069
 - createSession, 1070
 - destroy, 1070
 - getConnectionFactory, 1070
 - getResourceLifecycleManager, 1070
 - getSessionAcknowledgeMode, 1070
 - init, 1071
 - operator=, 1071
 - setConnectionFactory, 1071
 - setSessionAcknowledgeMode, 1071
- activemq::cmsutil::CmsDestinationAccessor, 1071
 - ~CmsDestinationAccessor, 1073
 - checkDestinationResolver, 1073
 - CmsDestinationAccessor, 1073
 - destroy, 1073
 - getDestinationResolver, 1073
 - init, 1073
 - isPubSubDomain, 1073
 - operator=, 1074
 - resolveDestinationName, 1074
 - setDestinationResolver, 1074
 - setPubSubDomain, 1074
- activemq::cmsutil::CmsTemplate, 1083
 - ~CmsTemplate, 1087
 - CmsTemplate, 1087

- DEFAULT_PRIORITY, 1096
- DEFAULT_TIME_TO_LIVE, 1096
- destroy, 1087
- execute, 1087, 1088
- getDefaultDestination, 1088, 1089
- getDefaultDestinationName, 1089
- getDeliveryMode, 1089
- getPriority, 1089
- getReceiveTimeout, 1089
- getTimeToLive, 1089
- init, 1089
- isExplicitQosEnabled, 1090
- isMessageIdEnabled, 1090
- isMessageTimestampEnabled, 1090
- isNoLocal, 1090
- operator=, 1090
- ProducerExecutor, 1096
- receive, 1090, 1091
- RECEIVE_TIMEOUT_INDEFINITE_WAIT, 1096
- RECEIVE_TIMEOUT_NO_WAIT, 1096
- ReceiveExecutor, 1096
- receiveSelected, 1091, 1092
- ResolveProducerExecutor, 1096
- ResolveReceiveExecutor, 1096
- send, 1092, 1093
- SendExecutor, 1096
- setDefaultDestination, 1093
- setDefaultDestinationName, 1093
- setDeliveryMode, 1093
- setDeliveryPersistent, 1094
- setExplicitQosEnabled, 1094
- setMessageIdEnabled, 1094
- setMessageTimestampEnabled, 1095
- setNoLocal, 1095
- setPriority, 1095
- setPubSubDomain, 1095
- setReceiveTimeout, 1095
- setTimeToLive, 1095
- activemq::cmsutil::CmsTemplate::ProducerExecutor, 2867
 - ~ProducerExecutor, 2868
 - action, 2869
 - destination, 2869
 - doInCms, 2868
 - getDestination, 2868
 - operator=, 2869
 - parent, 2869
 - ProducerExecutor, 2868
- activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2970
 - ~ReceiveExecutor, 2971
 - destination, 2972
 - doInCms, 2971
 - getDestination, 2972
 - getMessage, 2972
 - message, 2972
 - noLocal, 2972
 - operator=, 2972
 - parent, 2972
 - ReceiveExecutor, 2971
 - selector, 2972
- activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3070
 - ~ResolveProducerExecutor, 3071
 - getDestination, 3071
 - operator=, 3071
 - ResolveProducerExecutor, 3071
- activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3071
 - ~ResolveReceiveExecutor, 3072
 - getDestination, 3072
 - operator=, 3072
 - ResolveReceiveExecutor, 3072
- activemq::cmsutil::CmsTemplate::SendExecutor, 3135
 - ~SendExecutor, 3136
 - doInCms, 3136
 - operator=, 3136
 - SendExecutor, 3136
- activemq::cmsutil::DestinationResolver, 1642
 - ~DestinationResolver, 1642
 - destroy, 1642
 - init, 1643
 - resolveDestinationName, 1643
- activemq::cmsutil::DynamicDestinationResolver, 1704
 - ~DynamicDestinationResolver, 1705
 - destroy, 1705
 - DynamicDestinationResolver, 1705
 - init, 1705
 - operator=, 1705
 - resolveDestinationName, 1705
- activemq::cmsutil::MessageCreator, 2426
 - ~MessageCreator, 2426
 - createMessage, 2426
- activemq::cmsutil::PooledSession, 2765
 - ~PooledSession, 2767
 - close, 2768
 - commit, 2768
 - createBrowser, 2768
 - createBytesMessage, 2769
 - createCachedConsumer, 2769
 - createCachedProducer, 2770
 - createConsumer, 2770, 2771
 - createDurableConsumer, 2771
 - createMapMessage, 2772
 - createMessage, 2772

- createProducer, 2772
- createQueue, 2773
- createStreamMessage, 2773
- createTemporaryQueue, 2773
- createTemporaryTopic, 2774
- createTextMessage, 2774
- createTopic, 2774
- getAcknowledgeMode, 2775
- getSession, 2775
- isTransacted, 2775
- operator=, 2776
- PooledSession, 2767
- recover, 2776
- rollback, 2776
- unsubscribe, 2776
- activemq::cmsutil::ProducerCallback, 2866
 - ~ProducerCallback, 2867
 - doInCms, 2867
- activemq::cmsutil::ResourceLifecycleManager, 3074
 - ~ResourceLifecycleManager, 3075
 - addConnection, 3075
 - addDestination, 3075
 - addMessageConsumer, 3075
 - addMessageProducer, 3075
 - addSession, 3075
 - destroy, 3076
 - operator=, 3076
 - releaseAll, 3076
 - ResourceLifecycleManager, 3075
- activemq::cmsutil::SessionCallback, 3160
 - ~SessionCallback, 3161
 - doInCms, 3161
- activemq::cmsutil::SessionPool, 3215
 - ~SessionPool, 3216
 - getResourceLifecycleManager, 3217
 - operator=, 3217
 - returnSession, 3217
 - SessionPool, 3216
 - takeSession, 3217
- activemq::commands, 85
- activemq::commands::ActiveMQBlobMessage, 166
 - ~ActiveMQBlobMessage, 167
 - ActiveMQBlobMessage, 167
 - BINARY_MIME_TYPE, 170
 - clone, 167
 - cloneDataStructure, 168
 - copyDataStructure, 168
 - equals, 168
 - getDataStructureType, 168
 - getMimeType, 168
 - getName, 169
 - getRemoteBlobUrl, 169
 - ID_ACTIVEMQBLOBMESSAGE, 170
 - isDeletedByBroker, 169
 - setDeletedByBroker, 169
 - setMimeType, 169
 - setName, 170
 - setRemoteBlobUrl, 170
 - toString, 170
- activemq::commands::ActiveMQBytesMessage, 194
 - ~ActiveMQBytesMessage, 198
 - ActiveMQBytesMessage, 198
 - clearBody, 198
 - clone, 198
 - cloneDataStructure, 198
 - copyDataStructure, 198
 - equals, 199
 - getBodyBytes, 199
 - getBodyLength, 199
 - getDataStructureType, 200
 - ID_ACTIVEMQBYTESMESSAGE, 209
 - onSend, 200
 - readBoolean, 200
 - readByte, 200
 - readBytes, 201
 - readChar, 202
 - readDouble, 202
 - readFloat, 202
 - readInt, 203
 - readLong, 203
 - readShort, 203
 - readString, 204
 - readUnsignedShort, 204
 - readUTF, 204
 - reset, 205
 - setBodyBytes, 205
 - toString, 205
 - writeBoolean, 206
 - writeByte, 206
 - writeBytes, 206
 - writeChar, 207
 - writeDouble, 207
 - writeFloat, 207
 - writeInt, 208
 - writeLong, 208
 - writeShort, 208
 - writeString, 208
 - writeUnsignedShort, 209
 - writeUTF, 209
- activemq::commands::ActiveMQDestination, 279
 - ~ActiveMQDestination, 282
 - ActiveMQDestination, 282
 - advisory, 288
 - ADVISORY_PREFIX, 288

- cloneDataStructure, 282
- COMPOSITE_SEPARATOR, 288
- CONNECTION_ADVISORY_PREFIX, 288
- CONSUMER_ADVISORY_PREFIX, 288
- copyDataStructure, 282
- createDestination, 282
- createTemporaryName, 283
- DEFAULT_ORDERED_TARGET, 288
- equals, 283
- exclusive, 288
- getClientId, 283
- getCMSDestination, 283
- getDataStructureType, 284
- getDestinationType, 284
- getOptions, 284
- getOrderedTarget, 284
- getPhysicalName, 284, 285
- ID_ACTIVEMQDESTINATION, 289
- isAdvisory, 285
- isComposite, 285
- isConnectionAdvisory, 285
- isConsumerAdvisory, 285
- isExclusive, 285
- isOrdered, 286
- isProducerAdvisory, 286
- isQueue, 286
- isTemporary, 286
- isTopic, 286
- isWildcard, 286
- options, 289
- ordered, 289
- orderedTarget, 289
- physicalName, 289
- PRODUCER_ADVISORY_PREFIX, 289
- QUEUE_QUALIFIED_PREFIX, 289
- setAdvisory, 287
- setExclusive, 287
- setOrdered, 287
- setOrderedTarget, 287
- setPhysicalName, 287
- TEMP_POSTFIX, 289
- TEMP_PREFIX, 289
- TEMP_QUEUE_QUALIFIED_PREFIX, 289
- TEMP_TOPIC_QUALIFIED_PREFIX, 289
- TOPIC_QUALIFIED_PREFIX, 289
- toString, 287
- activemq::commands::ActiveMQDestination::DestinationFilter, 1613
 - ANY_CHILD, 1613
 - ANY_DESCENDENT, 1613
- activemq::commands::ActiveMQMapMessage, 316
 - ~ActiveMQMapMessage, 319
 - ActiveMQMapMessage, 319
 - beforeMarshal, 319
 - checkMapIsUnmarshalled, 319
 - clearBody, 319
 - clone, 319
 - cloneDataStructure, 319
 - copyDataStructure, 320
 - equals, 320
 - getBoolean, 320
 - getByte, 320
 - getBytes, 321
 - getChar, 321
 - getDataStructureType, 321
 - getDouble, 322
 - getFloat, 322
 - getInt, 322
 - getLong, 323
 - getMap, 323
 - getMapNames, 323
 - getShort, 323
 - getString, 324
 - ID_ACTIVEMQMAPMESSAGE, 328
 - isMarshalAware, 324
 - itemExists, 324
 - setBoolean, 325
 - setByte, 325
 - setBytes, 325
 - setChar, 326
 - setDouble, 326
 - setFloat, 326
 - setInt, 327
 - setLong, 327
 - setShort, 327
 - setString, 327
 - toString, 328
- activemq::commands::ActiveMQMessage, 352
 - ~ActiveMQMessage, 353
 - ActiveMQMessage, 353
 - clone, 353
 - cloneDataStructure, 353
 - copyDataStructure, 354
 - equals, 354
 - getDataStructureType, 354
 - ID_ACTIVEMQMESSAGE, 355
 - toString, 354
- activemq::commands::ActiveMQMessageTemplate, 379
 - ~ActiveMQMessageTemplate, 383
 - acknowledge, 383
 - ActiveMQMessageTemplate, 383
 - clearBody, 383

- clearProperties, 383
- equals, 383
- failIfReadOnlyBody, 384
- failIfReadOnlyProperties, 384
- failIfWriteOnlyBody, 384
- getBooleanProperty, 384
- getByteProperty, 384
- getCMSCorrelationID, 385
- getCMSDeliveryMode, 385
- getCMSDestination, 385
- getCMSExpiration, 385
- getCMSMessageID, 386
- getCMSPriority, 386
- getCMSRedelivered, 386
- getCMSReplyTo, 387
- getCMSTimestamp, 387
- getCMSType, 387
- getDoubleProperty, 387
- getFloatProperty, 388
- getIntProperty, 388
- getLongProperty, 389
- getPropertyNames, 389
- getShortProperty, 389
- getStringProperty, 389
- onSend, 390
- propertyExists, 390
- setBooleanProperty, 390
- setByteProperty, 391
- setCMSCorrelationID, 391
- setCMSDeliveryMode, 391
- setCMSDestination, 392
- setCMSExpiration, 392
- setCMSMessageID, 392
- setCMSPriority, 392
- setCMSRedelivered, 393
- setCMSReplyTo, 393
- setCMSTimestamp, 393
- setCMSType, 394
- setDoubleProperty, 394
- setFloatProperty, 394
- setIntProperty, 395
- setLongProperty, 395
- setShortProperty, 395
- setStringProperty, 396
- activemq::commands::ActiveMQObjectMessage, 396
 - ~ActiveMQObjectMessage, 397
 - ActiveMQObjectMessage, 397
 - clone, 397
 - cloneDataStructure, 397
 - copyDataStructure, 398
 - equals, 398
 - getDataStructureType, 398
 - ID_ACTIVEMQOBJECTMESSAGE, 399
 - toString, 398
- activemq::commands::ActiveMQQueue, 435
 - ~ActiveMQQueue, 436
 - ActiveMQQueue, 436
 - clone, 436
 - cloneDataStructure, 436
 - copy, 436
 - copyDataStructure, 437
 - equals, 437
 - getCMSDestination, 437
 - getCMSProperties, 437
 - getDataStructureType, 437
 - getDestinationType, 438
 - getQueueName, 438
 - ID_ACTIVEMQQUEUE, 438
 - toString, 438
- activemq::commands::ActiveMQStreamMessage, 485
 - ~ActiveMQStreamMessage, 489
 - ActiveMQStreamMessage, 489
 - clearBody, 489
 - clone, 489
 - cloneDataStructure, 489
 - copyDataStructure, 489
 - equals, 489
 - getDataStructureType, 490
 - ID_ACTIVEMQSTREAMMESSAGE, 499
 - onSend, 490
 - readBoolean, 490
 - readByte, 490
 - readBytes, 491
 - readChar, 492
 - readDouble, 492
 - readFloat, 493
 - readInt, 493
 - readLong, 493
 - readShort, 494
 - readString, 494
 - readUnsignedShort, 495
 - reset, 495
 - toString, 495
 - writeBoolean, 495
 - writeByte, 496
 - writeBytes, 496
 - writeChar, 497
 - writeDouble, 497
 - writeFloat, 497
 - writeInt, 498
 - writeLong, 498
 - writeShort, 498
 - writeString, 498
 - writeUnsignedShort, 499

- activemq::commands::ActiveMQTempDestination, 523
 - ~ActiveMQTempDestination, 525
 - ActiveMQTempDestination, 525
 - cloneDataStructure, 525
 - close, 525
 - connection, 526
 - copyDataStructure, 525
 - equals, 525
 - getDataStructureType, 526
 - ID_ACTIVEMQTEMPDESTINATION, 526
 - setConnection, 526
 - toString, 526
- activemq::commands::ActiveMQTempQueue, 549
 - ~ActiveMQTempQueue, 550
 - ActiveMQTempQueue, 550
 - clone, 550
 - cloneDataStructure, 550
 - copy, 550
 - copyDataStructure, 550
 - destroy, 551
 - equals, 551
 - getCMSDestination, 551
 - getCMSProperties, 551
 - getDataStructureType, 551
 - getDestinationType, 552
 - getQueueName, 552
 - ID_ACTIVEMQTEMPQUEUE, 552
 - toString, 552
- activemq::commands::ActiveMQTempTopic, 576
 - ~ActiveMQTempTopic, 578
 - ActiveMQTempTopic, 578
 - clone, 578
 - cloneDataStructure, 578
 - copy, 578
 - copyDataStructure, 578
 - destroy, 579
 - equals, 579
 - getCMSDestination, 579
 - getCMSProperties, 579
 - getDataStructureType, 579
 - getDestinationType, 580
 - getTopicName, 580
 - ID_ACTIVEMQTEMPTOPIC, 580
 - toString, 580
- activemq::commands::ActiveMQTextMessage, 604
 - ~ActiveMQTextMessage, 606
 - ActiveMQTextMessage, 606
 - beforeMarshal, 606
 - clearBody, 606
 - clone, 606
 - cloneDataStructure, 606
 - copyDataStructure, 607
 - equals, 607
 - getDataStructureType, 607
 - getSize, 607
 - getText, 608
 - ID_ACTIVEMQTEXTMESSAGE, 609
 - setText, 608
 - text, 609
 - toString, 608
- activemq::commands::ActiveMQTopic, 633
 - ~ActiveMQTopic, 634
 - ActiveMQTopic, 634
 - clone, 634
 - cloneDataStructure, 634
 - copy, 634
 - copyDataStructure, 635
 - equals, 635
 - getCMSDestination, 635
 - getCMSProperties, 635
 - getDataStructureType, 635
 - getDestinationType, 636
 - getTopicName, 636
 - ID_ACTIVEMQTOPIC, 636
 - toString, 636
- activemq::commands::BaseCommand, 694
 - ~BaseCommand, 696
 - BaseCommand, 696
 - copyDataStructure, 696
 - equals, 696
 - getCommandId, 697
 - isBrokerInfo, 697
 - isConnectionInfo, 698
 - isConsumerInfo, 698
 - isKeepAliveInfo, 698
 - isMessage, 698
 - isMessageAck, 698
 - isMessageDispatch, 698
 - isMessageDispatchNotification, 698
 - isProducerAck, 699
 - isProducerInfo, 699
 - isRemoveInfo, 699
 - isRemoveSubscriptionInfo, 699
 - isResponse, 699
 - isResponseRequired, 699
 - isShutdownInfo, 700
 - isTransactionInfo, 700
 - isWireFormatInfo, 700
 - setCommandId, 700
 - setResponseRequired, 700
 - toString, 700
- activemq::commands::BaseDataStructure, 764
 - ~BaseDataStructure, 765

- afterMarshal, 765
- afterUnmarshal, 765
- beforeMarshal, 765
- beforeUnmarshal, 765
- copyDataStructure, 766
- equals, 766
- getMarshaledForm, 766
- isMarshalAware, 766
- setMarshaledForm, 767
- toString, 767
- activemq::commands::BooleanExpression, 786
 - ~BooleanExpression, 786
 - BooleanExpression, 786
 - cloneDataStructure, 786
 - copyDataStructure, 786
 - equals, 787
 - toString, 787
- activemq::commands::BrokerError, 793
 - ~BrokerError, 794
 - BrokerError, 794
 - cloneDataStructure, 794
 - copyDataStructure, 794
 - getCause, 794
 - getDataStructureType, 795
 - getExceptionClass, 795
 - getMessage, 795
 - getStackTraceElements, 795
 - setCause, 795
 - setExceptionClass, 796
 - setMessage, 796
 - setStackTraceElements, 796
 - visit, 796
- activemq::commands::BrokerError::StackTraceElement, 3350
 - 3350
 - ClassName, 3351
 - FileName, 3351
 - LineNumber, 3351
 - MethodName, 3351
- activemq::commands::BrokerId, 798
 - ~BrokerId, 799
 - BrokerId, 799
 - cloneDataStructure, 799
 - COMPARATOR, 799
 - compareTo, 799
 - copyDataStructure, 799
 - equals, 800
 - getDataStructureType, 800
 - getValue, 800
 - ID_BROKERID, 801
 - operator<, 800
 - operator=, 800
 - operator==, 800
 - setValue, 800
 - toString, 800
 - value, 801
- activemq::commands::BrokerInfo, 824
 - ~BrokerInfo, 826
 - brokerId, 831
 - BrokerInfo, 826
 - brokerName, 831
 - brokerUploadUrl, 831
 - brokerURL, 831
 - cloneDataStructure, 826
 - connectionId, 831
 - copyDataStructure, 826
 - duplexConnection, 831
 - equals, 826
 - faultTolerantConfiguration, 831
 - getBrokerId, 827
 - getBrokerName, 827
 - getBrokerUploadUrl, 827
 - getBrokerURL, 827
 - getConnectionId, 827
 - getDataStructureType, 827
 - getNetworkProperties, 828
 - getPeerBrokerInfos, 828
 - ID_BROKERINFO, 831
 - isBrokerInfo, 828
 - isDuplexConnection, 828
 - isFaultTolerantConfiguration, 829
 - isMasterBroker, 829
 - isNetworkConnection, 829
 - isSlaveBroker, 829
 - masterBroker, 831
 - networkConnection, 831
 - networkProperties, 831
 - peerBrokerInfos, 831
 - setBrokerId, 829
 - setBrokerName, 829
 - setBrokerUploadUrl, 829
 - setBrokerURL, 829
 - setConnectionId, 829
 - setDuplexConnection, 829
 - setFaultTolerantConfiguration, 829
 - setMasterBroker, 829
 - setNetworkConnection, 829
 - setNetworkProperties, 829
 - setPeerBrokerInfos, 829
 - setSlaveBroker, 829
 - slaveBroker, 831
 - toString, 829
 - visit, 830
- activemq::commands::Command, 1107
 - ~Command, 1108
 - getCommandId, 1108
 - isBrokerInfo, 1108
 - isConnectionInfo, 1109
 - isConsumerInfo, 1109

- isKeepAliveInfo, 1109
- isMessage, 1109
- isMessageAck, 1109
- isMessageDispatch, 1109
- isMessageDispatchNotification, 1109
- isProducerAck, 1109
- isProducerInfo, 1110
- isRemoveInfo, 1110
- isRemoveSubscriptionInfo, 1110
- isResponse, 1110
- isResponseRequired, 1110
- isShutdownInfo, 1110
- isTransactionInfo, 1110
- isWireFormatInfo, 1111
- setCommandId, 1111
- setResponseRequired, 1111
- toString, 1111
- visit, 1112
- activemq::commands::ConnectionControl, 1172
 - ~ConnectionControl, 1173
 - cloneDataStructure, 1173
 - close, 1176
 - connectedBrokers, 1176
 - ConnectionControl, 1173
 - copyDataStructure, 1173
 - equals, 1173
 - exit, 1176
 - faultTolerant, 1176
 - getConnectedBrokers, 1174
 - getDataStructureType, 1174
 - getReconnectTo, 1174, 1175
 - ID_CONNECTIONCONTROL, 1176
 - isClose, 1175
 - isExit, 1175
 - isFaultTolerant, 1175
 - isRebalanceConnection, 1175
 - isResume, 1175
 - isSuspend, 1175
 - rebalanceConnection, 1176
 - reconnectTo, 1176
 - resume, 1176
 - setClose, 1175
 - setConnectedBrokers, 1175
 - setExit, 1175
 - setFaultTolerant, 1175
 - setRebalanceConnection, 1175
 - setReconnectTo, 1175
 - setResume, 1175
 - setSuspend, 1175
 - suspend, 1176
 - toString, 1175
 - visit, 1176
- activemq::commands::ConnectionError, 1200
 - ~ConnectionError, 1202
 - cloneDataStructure, 1202
 - ConnectionError, 1202
 - connectionId, 1204
 - copyDataStructure, 1202
 - equals, 1202
 - exception, 1204
 - getConnectionId, 1202, 1203
 - getDataStructureType, 1203
 - getException, 1203
 - ID_CONNECTIONERROR, 1204
 - setConnectionId, 1203
 - setException, 1203
 - toString, 1203
 - visit, 1203
- activemq::commands::ConnectionId, 1231
 - ~ConnectionId, 1232
 - cloneDataStructure, 1232
 - COMPARATOR, 1232
 - compareTo, 1232
 - ConnectionId, 1232
 - copyDataStructure, 1232
 - equals, 1232, 1233
 - getDataStructureType, 1233
 - getValue, 1233
 - ID_CONNECTIONID, 1234
 - operator<, 1233
 - operator=, 1233
 - operator==, 1233
 - setValue, 1233
 - toString, 1233
 - value, 1234
- activemq::commands::ConnectionInfo, 1257
 - ~ConnectionInfo, 1259
 - brokerMasterConnector, 1263
 - brokerPath, 1263
 - clientId, 1263
 - clientMaster, 1263
 - cloneDataStructure, 1259
 - connectionId, 1263
 - ConnectionInfo, 1259
 - copyDataStructure, 1259
 - createRemoveCommand, 1259
 - equals, 1260
 - faultTolerant, 1263
 - getBrokerPath, 1260
 - getClientId, 1260
 - getConnectionId, 1260
 - getDataStructureType, 1260
 - getPassword, 1260, 1261
 - getUserName, 1261
 - ID_CONNECTIONINFO, 1263
 - isBrokerMasterConnector, 1261
 - isClientMaster, 1261
 - isConnectionInfo, 1261

- isFaultTolerant, 1261
- isManageable, 1262
- manageable, 1263
- password, 1263
- setBrokerMasterConnector, 1262
- setBrokerPath, 1262
- setClientId, 1262
- setClientMaster, 1262
- setConnectionId, 1262
- setFaultTolerant, 1262
- setManageable, 1262
- setPassword, 1262
- setUserName, 1262
- toString, 1262
- userName, 1263
- visit, 1262
- activemq::commands::ConsumerControl, 1302
 - ~ConsumerControl, 1303
 - cloneDataStructure, 1303
 - close, 1306
 - ConsumerControl, 1303
 - consumerId, 1306
 - copyDataStructure, 1303
 - destination, 1306
 - equals, 1304
 - flush, 1306
 - getConsumerId, 1304
 - getDataStructureType, 1304
 - getDestination, 1304, 1305
 - getPrefetch, 1305
 - ID_CONSUMERCONTROL, 1306
 - isClose, 1305
 - isFlush, 1305
 - isStart, 1305
 - isStop, 1305
 - prefetch, 1306
 - setClose, 1305
 - setConsumerId, 1305
 - setDestination, 1305
 - setFlush, 1305
 - setPrefetch, 1305
 - setStart, 1305
 - setStop, 1305
 - start, 1306
 - stop, 1306
 - toString, 1305
 - visit, 1306
- activemq::commands::ConsumerId, 1330
 - ~ConsumerId, 1332
 - cloneDataStructure, 1332
 - COMPARATOR, 1332
 - compareTo, 1332
 - connectionId, 1334
 - ConsumerId, 1332
 - copyDataStructure, 1332
 - equals, 1332
 - getConnectionId, 1333
 - getDataStructureType, 1333
 - getParentId, 1333
 - getSessionId, 1333
 - getValue, 1333
 - ID_CONSUMERID, 1334
 - operator<, 1333
 - operator=, 1333
 - operator==, 1333
 - sessionId, 1334
 - setConnectionId, 1333
 - setSessionId, 1333
 - setValue, 1333
 - toString, 1333
 - value, 1334
- activemq::commands::ConsumerInfo, 1357
 - ~ConsumerInfo, 1360
 - additionalPredicate, 1365
 - brokerPath, 1365
 - browser, 1365
 - cloneDataStructure, 1360
 - consumerId, 1365
 - ConsumerInfo, 1360
 - copyDataStructure, 1360
 - createRemoveCommand, 1360
 - destination, 1365
 - dispatchAsync, 1365
 - equals, 1360
 - exclusive, 1365
 - getAdditionalPredicate, 1360, 1361
 - getBrokerPath, 1361
 - getConsumerId, 1361
 - getDataStructureType, 1361
 - getDestination, 1361, 1362
 - getMaximumPendingMessageLimit, 1362
 - getNetworkConsumerPath, 1362
 - getPrefetchSize, 1362
 - getPriority, 1362
 - getSelector, 1362
 - getSubscriptionName, 1362
 - ID_CONSUMERINFO, 1365
 - isBrowser, 1362
 - isConsumerInfo, 1362
 - isDispatchAsync, 1362
 - isExclusive, 1363
 - isNetworkSubscription, 1363
 - isNoLocal, 1363
 - isNoRangeAcks, 1363
 - isOptimizedAcknowledge, 1363
 - isRetroactive, 1363
 - maximumPendingMessageLimit, 1365
 - networkConsumerPath, 1365

- networkSubscription, 1365
- noLocal, 1365
- noRangeAcks, 1365
- optimizedAcknowledge, 1365
- prefetchSize, 1365
- priority, 1365
- retroactive, 1365
- selector, 1365
- setAdditionalPredicate, 1363
- setBrokerPath, 1363
- setBrowser, 1363
- setConsumerId, 1363
- setDestination, 1363
- setDispatchAsync, 1363
- setExclusive, 1363
- setMaximumPendingMessageLimit, 1363
- setNetworkConsumerPath, 1363
- setNetworkSubscription, 1363
- setNoLocal, 1363
- setNoRangeAcks, 1363
- setOptimizedAcknowledge, 1363
- setPrefetchSize, 1363
- setPriority, 1363
- setRetroactive, 1363
- setSelector, 1363
- setSubscriptionName, 1363
- subscriptionName, 1365
- toString, 1363
- visit, 1364
- activemq::commands::ControlCommand, 1390
 - ~ControlCommand, 1391
 - cloneDataStructure, 1391
 - command, 1393
 - ControlCommand, 1391
 - copyDataStructure, 1391
 - equals, 1391
 - getCommand, 1392
 - getDataStructureType, 1392
 - ID_CONTROLCOMMAND, 1393
 - setCommand, 1392
 - toString, 1392
 - visit, 1392
- activemq::commands::DataArrayResponse, 1423
 - ~DataArrayResponse, 1424
 - cloneDataStructure, 1424
 - copyDataStructure, 1424
 - data, 1425
 - DataArrayResponse, 1424
 - equals, 1424
 - getData, 1425
 - getDataStructureType, 1425
 - ID_DATAARRAYRESPONSE, 1425
 - setData, 1425
 - toString, 1425
- activemq::commands::DataResponse, 1476
 - ~DataResponse, 1478
 - cloneDataStructure, 1478
 - copyDataStructure, 1478
 - data, 1479
 - DataResponse, 1478
 - equals, 1478
 - getData, 1478, 1479
 - getDataStructureType, 1479
 - ID_DATARESPONSE, 1479
 - setData, 1479
 - toString, 1479
- activemq::commands::DataStructure, 1553
 - ~DataStructure, 1554
 - cloneDataStructure, 1554
 - copyDataStructure, 1555
 - equals, 1556
 - getDataStructureType, 1557
 - toString, 1558
- activemq::commands::DestinationInfo, 1613
 - ~DestinationInfo, 1615
 - brokerPath, 1618
 - cloneDataStructure, 1615
 - connectionId, 1618
 - copyDataStructure, 1615
 - destination, 1618
 - DestinationInfo, 1615
 - equals, 1615
 - getBrokerPath, 1615, 1616
 - getConnectionId, 1616
 - getDataStructureType, 1616
 - getDestination, 1616, 1617
 - getOperationType, 1617
 - getTimeout, 1617
 - ID_DESTINATIONINFO, 1618
 - operationType, 1618
 - setBrokerPath, 1617
 - setConnectionId, 1617
 - setDestination, 1617
 - setOperationType, 1617
 - setTimeout, 1617
 - timeout, 1618
 - toString, 1617
 - visit, 1617
- activemq::commands::DiscoveryEvent, 1643
 - ~DiscoveryEvent, 1645
 - brokerName, 1647
 - cloneDataStructure, 1645
 - copyDataStructure, 1645
 - DiscoveryEvent, 1645
 - equals, 1645
 - getBrokerName, 1645, 1646
 - getDataStructureType, 1646

- getServiceName, 1646
- ID_DISCOVERYEVENT, 1647
- serviceName, 1647
- setBrokerName, 1646
- setServiceName, 1646
- toString, 1646
- activemq::commands::ExceptionResponse, 1720
 - ~ExceptionResponse, 1721
 - cloneDataStructure, 1721
 - copyDataStructure, 1721
 - equals, 1721
 - exception, 1722
 - ExceptionResponse, 1721
 - getDataStructureType, 1721
 - getException, 1722
 - ID_EXCEPTIONRESPONSE, 1722
 - setException, 1722
 - toString, 1722
- activemq::commands::FlushCommand, 1812
 - ~FlushCommand, 1813
 - cloneDataStructure, 1813
 - copyDataStructure, 1813
 - equals, 1813
 - FlushCommand, 1813
 - getDataStructureType, 1813
 - ID_FLUSHCOMMAND, 1814
 - toString, 1814
 - visit, 1814
- activemq::commands::IntegerResponse, 1956
 - ~IntegerResponse, 1957
 - cloneDataStructure, 1957
 - copyDataStructure, 1957
 - equals, 1957
 - getDataStructureType, 1957
 - getResult, 1958
 - ID_INTEGERRESPONSE, 1958
 - IntegerResponse, 1957
 - result, 1958
 - setResult, 1958
 - toString, 1958
- activemq::commands::JournalQueueAck, 2014
 - ~JournalQueueAck, 2015
 - cloneDataStructure, 2015
 - copyDataStructure, 2015
 - destination, 2017
 - equals, 2015
 - getDataStructureType, 2016
 - getDestination, 2016
 - getMessageAck, 2016
 - ID_JOURNALQUEUEACK, 2017
 - JournalQueueAck, 2015
 - messageAck, 2017
 - setDestination, 2016
 - setMessageAck, 2016
 - toString, 2016
- activemq::commands::JournalTopicAck, 2040
 - ~JournalTopicAck, 2042
 - clientId, 2045
 - cloneDataStructure, 2042
 - copyDataStructure, 2042
 - destination, 2045
 - equals, 2042
 - getClientId, 2042, 2043
 - getDataStructureType, 2043
 - getDestination, 2043, 2044
 - getMessageId, 2044
 - getMessageSequenceId, 2044
 - getSubscriptionName, 2044
 - getTransactionId, 2044
 - ID_JOURNALTOPICACK, 2045
 - JournalTopicAck, 2042
 - messageId, 2045
 - messageSequenceId, 2045
 - setClientId, 2044
 - setDestination, 2044
 - setMessageId, 2044
 - setMessageSequenceId, 2044
 - setSubscriptionName, 2044
 - setTransactionId, 2044
 - subscriptionName, 2045
 - toString, 2044
 - transactionId, 2045
- activemq::commands::JournalTrace, 2069
 - ~JournalTrace, 2070
 - cloneDataStructure, 2070
 - copyDataStructure, 2070
 - equals, 2071
 - getDataStructureType, 2071
 - getMessage, 2071
 - ID_JOURNALTRACE, 2072
 - JournalTrace, 2070
 - message, 2072
 - setMessage, 2071
 - toString, 2071
- activemq::commands::JournalTransaction, 2095
 - ~JournalTransaction, 2096
 - cloneDataStructure, 2096
 - copyDataStructure, 2097
 - equals, 2097
 - getDataStructureType, 2097
 - getTransactionId, 2097, 2098
 - getType, 2098
 - getWasPrepared, 2098
 - ID_JOURNALTRANSACTION, 2098
 - JournalTransaction, 2096
 - setTransactionId, 2098
 - setType, 2098
 - setWasPrepared, 2098

- toString, 2098
- transactionId, 2098
- type, 2098
- wasPrepared, 2098
- activemq::commands::KeepAliveInfo, 2122
 - ~KeepAliveInfo, 2123
 - cloneDataStructure, 2123
 - copyDataStructure, 2123
 - equals, 2124
 - getDataStructureType, 2124
 - ID_KEEPLIVEINFO, 2125
 - isKeepAliveInfo, 2124
 - KeepAliveInfo, 2123
 - toString, 2124
 - visit, 2124
- activemq::commands::LastPartialCommand, 2156
 - ~LastPartialCommand, 2157
 - cloneDataStructure, 2157
 - copyDataStructure, 2157
 - equals, 2157
 - getDataStructureType, 2157
 - ID_LASTPARTIALCOMMAND, 2158
 - LastPartialCommand, 2157
 - toString, 2158
- activemq::commands::LocalTransactionId, 2200
 - ~LocalTransactionId, 2202
 - cloneDataStructure, 2202
 - COMPARATOR, 2202
 - compareTo, 2202
 - connectionId, 2204
 - copyDataStructure, 2202
 - equals, 2202
 - getConnectionId, 2203
 - getDataStructureType, 2203
 - getValue, 2203
 - ID_LOCALTRANSACTIONID, 2204
 - LocalTransactionId, 2202
 - operator<, 2203
 - operator=, 2203
 - operator==, 2203
 - setConnectionId, 2203
 - setValue, 2203
 - toString, 2203
 - value, 2204
- activemq::commands::Message, 2358
 - ~Message, 2362
 - afterUnmarshal, 2362
 - arrival, 2374
 - beforeMarshal, 2362
 - brokerInTime, 2374
 - brokerOutTime, 2374
 - brokerPath, 2374
 - cloneDataStructure, 2362
 - cluster, 2374
 - compressed, 2374
 - connection, 2374
 - content, 2374
 - copyDataStructure, 2363
 - correlationId, 2374
 - dataStructure, 2374
 - DEFAULT_MESSAGE_SIZE, 2374
 - destination, 2374
 - droppable, 2374
 - equals, 2363
 - expiration, 2374
 - getAckHandler, 2364
 - getArrival, 2364
 - getBrokerInTime, 2364
 - getBrokerOutTime, 2364
 - getBrokerPath, 2364
 - getCluster, 2364
 - getConnection, 2364
 - getContent, 2364, 2365
 - getCorrelationId, 2365
 - getDataStructure, 2365
 - getDataStructureType, 2365
 - getDestination, 2365, 2366
 - getExpiration, 2366
 - getGroupID, 2366
 - getGroupSequence, 2366
 - getMarshaledProperties, 2366
 - getMessageId, 2366
 - getMessageProperties, 2366
 - getOriginalDestination, 2367
 - getOriginalTransactionId, 2367
 - getPriority, 2367
 - getProducerId, 2367
 - getRedeliveryCounter, 2367
 - getReplyTo, 2367
 - getSize, 2367
 - getTargetConsumerId, 2367, 2368
 - getTimestamp, 2368
 - getTransactionId, 2368
 - getType, 2368
 - getUserID, 2368
 - groupID, 2374
 - groupSequence, 2374
 - ID_MESSAGE, 2374
 - isCompressed, 2368
 - isDroppable, 2368
 - isExpired, 2368
 - isMarshalAware, 2368
 - isMessage, 2369
 - isPersistent, 2369
 - isReadOnlyBody, 2369
 - isReadOnlyProperties, 2369
 - isRecievedByDFBridge, 2369

- marshalledProperties, 2374
- Message, 2362
- messageId, 2374
- onSend, 2369
- originalDestination, 2374
- originalTransactionId, 2374
- persistent, 2374
- priority, 2374
- producerId, 2374
- recievedByDFBridge, 2374
- redeliveryCounter, 2374
- replyTo, 2374
- setAckHandler, 2369
- setArrival, 2370
- setBrokerInTime, 2370
- setBrokerOutTime, 2370
- setBrokerPath, 2370
- setCluster, 2370
- setCompressed, 2370
- setConnection, 2370
- setContent, 2370
- setCorrelationId, 2371
- setDataStructure, 2371
- setDestination, 2371
- setDroppable, 2371
- setExpiration, 2371
- setGroupID, 2371
- setGroupSequence, 2371
- setMarshaledProperties, 2371
- setMessageId, 2371
- setOriginalDestination, 2371
- setOriginalTransactionId, 2371
- setPersistent, 2371
- setPriority, 2371
- setProducerId, 2371
- setReadOnlyBody, 2371
- setReadOnlyProperties, 2372
- setRecievedByDFBridge, 2372
- setRedeliveryCounter, 2372
- setReplyTo, 2372
- setTargetConsumerId, 2372
- setTimestamp, 2372
- setTransactionId, 2372
- setType, 2372
- setUserID, 2372
- targetConsumerId, 2374
- timestamp, 2374
- toString, 2372
- transactionId, 2374
- type, 2374
- userId, 2374
- visit, 2373
- activemq::commands::MessageAck, 2394
 - ~MessageAck, 2395
- ackType, 2399
- cloneDataStructure, 2395
- consumerId, 2399
- copyDataStructure, 2395
- destination, 2399
- equals, 2395
- firstMessageId, 2399
- getAckType, 2396
- getConsumerId, 2396
- getDataStructureType, 2396
- getDestination, 2396, 2397
- getFirstMessageId, 2397
- getLastMessageId, 2397
- getMessageCount, 2397
- getTransactionId, 2397
- ID_MESSAGEACK, 2399
- isMessageAck, 2397
- lastMessageId, 2399
- MessageAck, 2395
- messageCount, 2399
- setAckType, 2397
- setConsumerId, 2398
- setDestination, 2398
- setFirstMessageId, 2398
- setLastMessageId, 2398
- setMessageCount, 2398
- setTransactionId, 2398
- toString, 2398
- transactionId, 2399
- visit, 2398
- activemq::commands::MessageDispatch, 2427
 - ~MessageDispatch, 2428
 - cloneDataStructure, 2428
 - consumerId, 2431
 - copyDataStructure, 2428
 - destination, 2431
 - equals, 2428
 - getConsumerId, 2429
 - getDataStructureType, 2429
 - getDestination, 2429
 - getMessage, 2429
 - getRedeliveryCounter, 2429
 - ID_MESSAGEDISPATCH, 2431
 - isMessageDispatch, 2429
 - message, 2431
 - MessageDispatch, 2428
 - redeliveryCounter, 2431
 - setConsumerId, 2429
 - setDestination, 2430
 - setMessage, 2430
 - setRedeliveryCounter, 2430
 - toString, 2430
 - visit, 2430

- activemq::commands::MessageDispatchNotification, 2462
 - ~MessageDispatchNotification, 2463
 - cloneDataStructure, 2463
 - consumerId, 2466
 - copyDataStructure, 2463
 - deliverySequenceId, 2466
 - destination, 2466
 - equals, 2464
 - getConsumerId, 2464
 - getDataStructureType, 2464
 - getDeliverySequenceId, 2464
 - getDestination, 2465
 - getMessageId, 2465
 - ID_MESSAGEDISPATCHNOTIFICATION, 2466
 - isMessageDispatchNotification, 2465
 - MessageDispatchNotification, 2463
 - messageId, 2466
 - setConsumerId, 2465
 - setDeliverySequenceId, 2465
 - setDestination, 2465
 - setMessageId, 2465
 - toString, 2465
 - visit, 2466
- activemq::commands::MessageId, 2494
 - ~MessageId, 2496
 - brokerSequenceId, 2499
 - cloneDataStructure, 2496
 - COMPARATOR, 2496
 - compareTo, 2496
 - copyDataStructure, 2496
 - equals, 2496, 2497
 - getBrokerSequenceId, 2497
 - getDataStructureType, 2497
 - getProducerId, 2497, 2498
 - getProducerSequenceId, 2498
 - ID_MESSAGEID, 2499
 - MessageId, 2496
 - operator<, 2498
 - operator=, 2498
 - operator==, 2498
 - producerId, 2499
 - producerSequenceId, 2499
 - setBrokerSequenceId, 2498
 - setProducerId, 2498
 - setProducerSequenceId, 2498
 - setTextView, 2498
 - setValue, 2498
 - toString, 2498
- activemq::commands::MessagePull, 2563
 - ~MessagePull, 2565
 - cloneDataStructure, 2565
 - consumerId, 2568
 - copyDataStructure, 2565
 - correlationId, 2568
 - destination, 2568
 - equals, 2565
 - getConsumerId, 2565, 2566
 - getCorrelationId, 2566
 - getDataStructureType, 2566
 - getDestination, 2566, 2567
 - getMessageId, 2567
 - getTimeout, 2567
 - ID_MESSAGEPULL, 2568
 - messageId, 2568
 - MessagePull, 2565
 - setConsumerId, 2567
 - setCorrelationId, 2567
 - setDestination, 2567
 - setMessageId, 2567
 - setTimeout, 2567
 - timeout, 2568
 - toString, 2567
 - visit, 2567
- activemq::commands::NetworkBridgeFilter, 2613
 - ~NetworkBridgeFilter, 2615
 - cloneDataStructure, 2615
 - copyDataStructure, 2615
 - equals, 2615
 - getDataStructureType, 2615
 - getNetworkBrokerId, 2616
 - getNetworkTTL, 2616
 - ID_NETWORKBRIDGEFILTER, 2616
 - NetworkBridgeFilter, 2615
 - networkBrokerId, 2616
 - networkTTL, 2616
 - setNetworkBrokerId, 2616
 - setNetworkTTL, 2616
 - toString, 2616
- activemq::commands::PartialCommand, 2727
 - ~PartialCommand, 2729
 - cloneDataStructure, 2729
 - commandId, 2731
 - copyDataStructure, 2729
 - data, 2731
 - equals, 2729
 - getCommandId, 2729
 - getData, 2730
 - getDataStructureType, 2730
 - ID_PARTIALCOMMAND, 2731
 - PartialCommand, 2729
 - setCommandId, 2730
 - setData, 2730
 - toString, 2730
- activemq::commands::ProducerAck, 2839
 - ~ProducerAck, 2840

- cloneDataStructure, 2840
- copyDataStructure, 2840
- equals, 2841
- getDataStructureType, 2841
- getProducerId, 2841
- getSize, 2841
- ID_PRODUCERACK, 2842
- isProducerAck, 2841
- ProducerAck, 2840
- producerId, 2842
- setProducerId, 2841
- setSize, 2842
- size, 2842
- toString, 2842
- visit, 2842
- activemq::commands::ProducerId, 2869
 - ~ProducerId, 2871
 - cloneDataStructure, 2871
 - COMPARATOR, 2871
 - compareTo, 2871
 - connectionId, 2873
 - copyDataStructure, 2871
 - equals, 2871, 2872
 - getConnectionId, 2872
 - getDataStructureType, 2872
 - getParentId, 2872
 - getSessionId, 2873
 - getValue, 2873
 - ID_PRODUCERID, 2873
 - operator<, 2873
 - operator=, 2873
 - operator==, 2873
 - ProducerId, 2871
 - sessionId, 2873
 - setConnectionId, 2873
 - setProducerSessionKey, 2873
 - setSessionId, 2873
 - setValue, 2873
 - toString, 2873
 - value, 2874
- activemq::commands::ProducerInfo, 2897
 - ~ProducerInfo, 2899
 - brokerPath, 2902
 - cloneDataStructure, 2899
 - copyDataStructure, 2899
 - createRemoveCommand, 2899
 - destination, 2902
 - dispatchAsync, 2902
 - equals, 2899
 - getBrokerPath, 2899, 2900
 - getDataStructureType, 2900
 - getDestination, 2900
 - getProducerId, 2900
 - getWindowSize, 2900
 - ID_PRODUCERINFO, 2902
 - isDispatchAsync, 2900
 - isProducerInfo, 2900
 - producerId, 2902
 - ProducerInfo, 2899
 - setBrokerPath, 2900
 - setDestination, 2901
 - setDispatchAsync, 2901
 - setProducerId, 2901
 - setWindowSize, 2901
 - toString, 2901
 - visit, 2901
 - windowSize, 2902
- activemq::commands::RemoveInfo, 2988
 - ~RemoveInfo, 2989
 - cloneDataStructure, 2989
 - copyDataStructure, 2989
 - equals, 2989
 - getDataStructureType, 2989
 - getLastDeliveredSequenceId, 2990
 - getObjectId, 2990
 - ID_REMOVEINFO, 2991
 - isRemoveInfo, 2990
 - lastDeliveredSequenceId, 2991
 - objectId, 2991
 - RemoveInfo, 2989
 - setLastDeliveredSequenceId, 2990
 - setObjectId, 2990
 - toString, 2990
 - visit, 2990
- activemq::commands::RemoveSubscriptionInfo, 3015
 - ~RemoveSubscriptionInfo, 3016
 - clientId, 3019
 - cloneDataStructure, 3016
 - connectionId, 3019
 - copyDataStructure, 3016
 - equals, 3016
 - getClientId, 3017
 - getConnectionId, 3017
 - getDataStructureType, 3017
 - getSubscriptionName, 3017, 3018
 - ID_REMOVESUBSCRIPTIONINFO, 3019
 - isRemoveSubscriptionInfo, 3018
 - RemoveSubscriptionInfo, 3016
 - setClientId, 3018
 - setConnectionId, 3018
 - setSubscriptionName, 3018
 - subscriptionName, 3019
 - toString, 3018
 - visit, 3018
- activemq::commands::ReplayCommand, 3043
 - ~ReplayCommand, 3044

- cloneDataStructure, 3044
- copyDataStructure, 3044
- equals, 3044
- firstNakNumber, 3046
- getDataStructureType, 3044
- getFirstNakNumber, 3045
- getLastNakNumber, 3045
- ID_REPLAYCOMMAND, 3046
- lastNakNumber, 3046
- ReplayCommand, 3044
- setFirstNakNumber, 3045
- setLastNakNumber, 3045
- toString, 3045
- visit, 3045
- activemq::commands::Response, 3076
 - ~Response, 3077
 - cloneDataStructure, 3077
 - copyDataStructure, 3077
 - correlationId, 3079
 - equals, 3078
 - getCorrelationId, 3078
 - getDataStructureType, 3078
 - ID_RESPONSE, 3079
 - isResponse, 3078
 - Response, 3077
 - setCorrelationId, 3079
 - toString, 3079
 - visit, 3079
- activemq::commands::SessionId, 3161
 - ~SessionId, 3163
 - cloneDataStructure, 3163
 - COMPARATOR, 3163
 - compareTo, 3163
 - connectionId, 3165
 - copyDataStructure, 3163
 - equals, 3163, 3164
 - getConnectionId, 3164
 - getDataStructureType, 3164
 - getParentId, 3164
 - getValue, 3164
 - ID_SESSIONID, 3165
 - operator<, 3164
 - operator=, 3164
 - operator==, 3164
 - SessionId, 3163
 - setConnectionId, 3164
 - setValue, 3164
 - toString, 3164
 - value, 3165
- activemq::commands::SessionInfo, 3189
 - ~SessionInfo, 3189
 - cloneDataStructure, 3189
 - copyDataStructure, 3190
 - createRemoveCommand, 3190
 - equals, 3190
 - getAckMode, 3190
 - getDataStructureType, 3190
 - getSessionId, 3190, 3191
 - ID_SESSIONINFO, 3191
 - sessionId, 3191
 - SessionInfo, 3189
 - setAckMode, 3191
 - setSessionId, 3191
 - toString, 3191
 - visit, 3191
- activemq::commands::ShutdownInfo, 3249
 - ~ShutdownInfo, 3250
 - cloneDataStructure, 3250
 - copyDataStructure, 3250
 - equals, 3251
 - getDataStructureType, 3251
 - ID_SHUTDOWNINFO, 3252
 - isShutdownInfo, 3251
 - ShutdownInfo, 3250
 - toString, 3251
 - visit, 3251
- activemq::commands::SubscriptionInfo, 3433
 - ~SubscriptionInfo, 3435
 - clientId, 3438
 - cloneDataStructure, 3435
 - copyDataStructure, 3435
 - destination, 3438
 - equals, 3435
 - getClientId, 3435, 3436
 - getDataStructureType, 3436
 - getDestination, 3436, 3437
 - getSelector, 3437
 - getSubscriptionName, 3437
 - getSubscribedDestination, 3437
 - ID_SUBSCRIPTIONINFO, 3438
 - selector, 3438
 - setClientId, 3437
 - setDestination, 3437
 - setSelector, 3437
 - setSubscriptionName, 3437
 - setSubscribedDestination, 3437
 - subscriptionName, 3438
 - subscribedDestination, 3438
 - SubscriptionInfo, 3435
 - toString, 3437
- activemq::commands::TransactionId, 3569
 - ~TransactionId, 3571
 - cloneDataStructure, 3571
 - COMPARATOR, 3571
 - compareTo, 3571
 - copyDataStructure, 3571
 - equals, 3571
 - getDataStructureType, 3572

- ID_TRANSACTIONID, 3572
- operator<, 3572
- operator=, 3572
- operator==, 3572
- toString, 3572
- TransactionId, 3571
- activemq::commands::TransactionInfo, 3594
 - ~TransactionInfo, 3596
 - cloneDataStructure, 3596
 - connectionId, 3598
 - copyDataStructure, 3596
 - equals, 3596
 - getConnectionId, 3596, 3597
 - getDataStructureType, 3597
 - getTransactionId, 3597
 - getType, 3597
 - ID_TRANSACTIONINFO, 3598
 - isTransactionInfo, 3597
 - setConnectionId, 3597
 - setTransactionId, 3597
 - setType, 3597
 - toString, 3597
 - transactionId, 3598
 - TransactionInfo, 3596
 - type, 3598
 - visit, 3598
- activemq::commands::WireFormatInfo, 3717
 - ~WireFormatInfo, 3720
 - afterUnmarshal, 3720
 - beforeMarshal, 3720
 - cloneDataStructure, 3720
 - copyDataStructure, 3721
 - equals, 3721
 - getCacheSize, 3721
 - getDataStructureType, 3721
 - getMagic, 3722
 - getMarshaledProperties, 3722
 - getMaxInactivityDuration, 3722
 - getMaxInactivityDurationInitialDelay, 3722
 - getProperties, 3722, 3723
 - getVersion, 3723
 - ID_WIREFORMATINFO, 3727
 - isCacheEnabled, 3723
 - isMarshalAware, 3723
 - isSizePrefixDisabled, 3723
 - isStackTraceEnabled, 3724
 - isTcpNoDelayEnabled, 3724
 - isTightEncodingEnabled, 3724
 - isValid, 3724
 - isWireFormatInfo, 3724
 - setCacheEnabled, 3724
 - setCacheSize, 3725
 - setMagic, 3725
 - setMarshaledProperties, 3725
 - setMaxInactivityDuration, 3725
 - setMaxInactivityDurationInitialDelay, 3725
 - setProperties, 3726
 - setSizePrefixDisabled, 3726
 - setStackTraceEnabled, 3726
 - setTcpNoDelayEnabled, 3726
 - setTightEncodingEnabled, 3726
 - setVersion, 3727
 - toString, 3727
 - visit, 3727
 - WireFormatInfo, 3720
- activemq::commands::XATransactionId, 3765
 - ~XATransactionId, 3766
 - branchQualifier, 3769
 - cloneDataStructure, 3766
 - COMPARATOR, 3766
 - compareTo, 3766
 - copyDataStructure, 3766
 - equals, 3766, 3767
 - formatId, 3769
 - getBranchQualifier, 3767
 - getDataStructureType, 3767
 - getFormatId, 3767
 - getGlobalTransactionId, 3768
 - globalTransactionId, 3769
 - ID_XATRANSACTIONID, 3769
 - operator<, 3768
 - operator=, 3768
 - operator==, 3768
 - setBranchQualifier, 3768
 - setFormatId, 3768
 - setGlobalTransactionId, 3768
 - toString, 3768
 - XATransactionId, 3766
- activemq::core, 86
- activemq::core::ActiveMQAckHandler, 165
 - ~ActiveMQAckHandler, 166
 - acknowledgeMessage, 166
- activemq::core::ActiveMQConnection, 233
 - ~ActiveMQConnection, 238
 - ActiveMQConnection, 238
 - addDispatcher, 238
 - addProducer, 239
 - addTransportListener, 239
 - close, 239
 - createSession, 239
 - destroyDestination, 240
 - fire, 241
 - getBrokerURL, 241
 - getClientID, 241
 - getCloseTimeout, 241
 - getConnectionId, 241
 - getConnectionInfo, 241
 - getExceptionListener, 242

- getMetaData, 242
- getNextLocalTransactionId, 242
- getNextSessionId, 242
- getNextTempDestinationId, 243
- getPassword, 243
- getPrefetchPolicy, 243
- getProducerWindowSize, 243
- getRedeliveryPolicy, 243
- getSendTimeout, 244
- getTransport, 244
- getUsername, 244
- isAlwaysSyncSend, 244
- isClosed, 244
- isDispatchAsync, 245
- isStarted, 245
- isTransportFailed, 245
- isUseAsyncSend, 245
- isUseCompression, 245
- onCommand, 245
- oneway, 246
- onException, 246
- removeDispatcher, 246
- removeProducer, 246
- removeSession, 246
- removeTransportListener, 247
- sendPullRequest, 247
- setAlwaysSyncSend, 247
- setBrokerURL, 247
- setClientID, 247
- setCloseTimeout, 248
- setDefaultClientId, 248
- setDispatchAsync, 248
- setExceptionListener, 248
- setPassword, 248
- setPrefetchPolicy, 248
- setProducerWindowSize, 249
- setRedeliveryPolicy, 249
- setSendTimeout, 249
- setTransportInterruptionProcessingComplete, 249
- setUseAsyncSend, 249
- setUseCompression, 250
- setUsername, 250
- start, 250
- stop, 250
- syncRequest, 250
- transportInterrupted, 251
- transportResumed, 251
- activemq::core::ActiveMQConnectionFactory, 251
 - ~ActiveMQConnectionFactory, 254
 - ActiveMQConnectionFactory, 254
 - createConnection, 254, 255
 - DEFAULT_URI, 261
 - getBrokerURL, 256
 - getClientId, 256
 - getCloseTimeout, 256
 - getExceptionListener, 256
 - getPassword, 257
 - getPrefetchPolicy, 257
 - getProducerWindowSize, 257
 - getRedeliveryPolicy, 257
 - getSendTimeout, 257
 - getUsername, 258
 - isAlwaysSyncSend, 258
 - isDispatchAsync, 258
 - isUseAsyncSend, 258
 - isUseCompression, 258
 - setAlwaysSyncSend, 258
 - setBrokerURL, 259
 - setClientId, 259
 - setCloseTimeout, 259
 - setDispatchAsync, 259
 - setExceptionListener, 259
 - setPassword, 260
 - setPrefetchPolicy, 260
 - setProducerWindowSize, 260
 - setRedeliveryPolicy, 260
 - setSendTimeout, 261
 - setUseAsyncSend, 261
 - setUseCompression, 261
 - setUsername, 261
- activemq::core::ActiveMQConnectionMetaData, 262
 - ~ActiveMQConnectionMetaData, 263
 - ActiveMQConnectionMetaData, 263
 - getCMSMajorVersion, 263
 - getCMSMinorVersion, 263
 - getCMSProviderName, 263
 - getCMSVersion, 264
 - getCMSXPropertyNames, 264
 - getProviderMajorVersion, 264
 - getProviderMinorVersion, 264
 - getProviderVersion, 265
- activemq::core::ActiveMQConstants, 265
 - ACK_TYPE_CONSUMED, 267
 - ACK_TYPE_DELIVERED, 267
 - ACK_TYPE_INDIVIDUAL, 267
 - ACK_TYPE_POISON, 267
 - ACK_TYPE_REDELIVERED, 267
 - AckType, 267
 - CONNECTION_ALWAYS_SYNC_SEND, 268
 - CONNECTION_CLOSE_TIMEOUT, 268
 - CONNECTION_DISPATCH_ASYNC, 268
 - CONNECTION_PRODUCER_WINDOW_SIZE, 268
 - CONNECTION_SEND_TIMEOUT, 268

- CONNECTION_USEASYNCSEND, 268
- CONNECTION_USECOMPRESSION, 268
- CONSUMER_DISPATCHASYNC, 267
- CONSUMER_EXCLUSIVE, 267
- CONSUMER_NOLOCAL, 267
- CONSUMER_PREFECTCHSIZE, 267
- CONSUMER_PRIORITY, 267
- CONSUMER_RETROACTIVE, 267
- CONSUMER_SELECTOR, 267
- CUNSUMER_-
 - MAXPENDINGMSGLIMIT, 267
- DESTINATION_ADD_OPERATION, 267
- DESTINATION_REMOVE_-
 - OPERATION, 267
- DestinationActions, 267
- DestinationOption, 267
- NUM_OPTIONS, 267
- NUM_PARAMS, 268
- PARAM_CLIENTID, 268
- PARAM_PASSWORD, 268
- PARAM_USERNAME, 268
- toDestinationOption, 268
- toString, 268
- toURIOption, 268
- TRANSACTION_STATE_BEGIN, 267
- TRANSACTION_STATE_-
 - COMMITONEPHASE, 267
- TRANSACTION_STATE_-
 - COMMITTWOPHASE, 267
- TRANSACTION_STATE_END, 267
- TRANSACTION_STATE_FORGET, 267
- TRANSACTION_STATE_PREPARE, 267
- TRANSACTION_STATE_RECOVER, 267
- TRANSACTION_STATE_ROLLBACK, 267
- TransactionState, 267
- URIParam, 267
- activemq::core::ActiveMQConstants::StaticInitializer, 3356
 - ~StaticInitializer, 3357
 - destOptionMap, 3357
 - destOptions, 3357
 - StaticInitializer, 3357
 - uriParams, 3357
 - uriParamsMap, 3357
- activemq::core::ActiveMQConsumer, 268
 - ~ActiveMQConsumer, 271
 - acknowledge, 271
 - ActiveMQConsumer, 271
 - afterMessageIsConsumed, 272
 - beforeMessageIsConsumed, 272
 - clearMessagesInProgress, 272
 - close, 272
 - commit, 272
 - deliverAcks, 272
 - dequeue, 273
 - dispatch, 273
 - doClose, 273
 - getConsumerId, 273
 - getConsumerInfo, 273
 - getLastDeliveredSequenceId, 274
 - getMessageAvailableCount, 274
 - getMessageListener, 274
 - getMessageSelector, 274
 - getRedeliveryPolicy, 274
 - inProgressClearRequired, 275
 - isClosed, 275
 - isSynchronizationRegistered, 275
 - iterate, 275
 - receive, 275
 - receiveNoWait, 276
 - rollback, 276
 - setLastDeliveredSequenceId, 276
 - setMessageListener, 276
 - setRedeliveryPolicy, 277
 - setSynchronizationRegistered, 277
 - start, 277
 - stop, 277
- activemq::core::ActiveMQProducer, 423
 - ~ActiveMQProducer, 425
 - ActiveMQProducer, 425
 - close, 425
 - getDeliveryMode, 425
 - getDisableMessageID, 425
 - getDisableMessageTimeStamp, 426
 - getPriority, 426
 - getProducerId, 426
 - getProducerInfo, 426
 - getSendTimeout, 426
 - getTimeToLive, 427
 - isClosed, 427
 - onProducerAck, 427
 - send, 427–429
 - setDeliveryMode, 429
 - setDisableMessageID, 429
 - setDisableMessageTimeStamp, 430
 - setPriority, 430
 - setSendTimeout, 430
 - setTimeToLive, 430
- activemq::core::ActiveMQQueueBrowser, 438
 - ~ActiveMQQueueBrowser, 440
 - ActiveMQQueueBrowser, 440
 - Browser, 441
 - close, 440

- getEnumeration, 440
- getMessageSelector, 440
- getQueue, 440
- hasMoreMessages, 440
- nextMessage, 440
- activemq::core::ActiveMQSession, 465
 - ~ActiveMQSession, 469
 - acknowledge, 469
 - ActiveMQSession, 469
 - ActiveMQSessionExecutor, 482
 - addConsumer, 469
 - addProducer, 469
 - clearMessagesInProgress, 470
 - close, 470
 - commit, 470
 - createBrowser, 470, 471
 - createBytesMessage, 471
 - createConsumer, 472
 - createDurableConsumer, 473
 - createMapMessage, 473
 - createMessage, 473
 - createProducer, 473
 - createQueue, 474
 - createStreamMessage, 474
 - createTemporaryQueue, 474
 - createTemporaryTopic, 474
 - createTextMessage, 475
 - createTopic, 475
 - deliverAcks, 475
 - dispatch, 476
 - doStartTransaction, 476
 - fire, 476
 - getAcknowledgeMode, 476
 - getConnection, 476
 - getExceptionListener, 476
 - getLastDeliveredSequenceId, 477
 - getNextConsumerId, 477
 - getNextProducerId, 477
 - getSessionId, 477
 - getSessionInfo, 477
 - getTransactionContext, 478
 - isAutoAcknowledge, 478
 - isClientAcknowledge, 478
 - isDupsOkAcknowledge, 478
 - isIndividualAcknowledge, 478
 - isStarted, 478
 - isTransacted, 478
 - oneway, 478
 - recover, 479
 - redispatch, 479
 - removeConsumer, 479
 - removeProducer, 480
 - rollback, 480
 - send, 480
 - setLastDeliveredSequenceId, 480
 - start, 481
 - stop, 481
 - syncRequest, 481
 - unsubscribe, 481
 - wakeup, 481
- activemq::core::ActiveMQSessionExecutor, 482
 - ~ActiveMQSessionExecutor, 483
 - ActiveMQSessionExecutor, 483
 - clear, 483
 - clearMessagesInProgress, 483
 - close, 483
 - execute, 484
 - executeFirst, 484
 - getUnconsumedMessages, 484
 - hasUnconsumedMessages, 484
 - isEmpty, 484
 - isRunning, 484
 - iterate, 485
 - start, 485
 - stop, 485
 - wakeup, 485
- activemq::core::ActiveMQTransactionContext, 660
 - ~ActiveMQTransactionContext, 662
 - ActiveMQTransactionContext, 661
 - addSynchronization, 662
 - begin, 662
 - commit, 662
 - getTransactionId, 662
 - isInTransaction, 662
 - removeSynchronization, 663
 - rollback, 663
- activemq::core::DispatchData, 1670
 - DispatchData, 1671
 - getConsumerId, 1671
 - getMessage, 1671
- activemq::core::Dispatcher, 1671
 - ~Dispatcher, 1672
 - dispatch, 1672
- activemq::core::MessageDispatchChannel, 2431
 - ~MessageDispatchChannel, 2433
 - clear, 2433
 - close, 2433
 - dequeue, 2433
 - dequeueNoWait, 2433
 - enqueue, 2434
 - enqueueFirst, 2434
 - isClosed, 2434
 - isEmpty, 2434
 - isRunning, 2434
 - lock, 2434
 - MessageDispatchChannel, 2433
 - notify, 2434

- notifyAll, 2435
 - peek, 2435
 - removeAll, 2435
 - size, 2435
 - start, 2436
 - stop, 2436
 - tryLock, 2436
 - unlock, 2436
 - wait, 2436, 2437
- activemq::core::policies, 87
- activemq::core::policies::DefaultPrefetchPolicy, 1565
 - ~DefaultPrefetchPolicy, 1567
 - clone, 1567
 - DEFAULT_DURABLE_TOPIC_PREFETCH, 1569
 - DEFAULT_QUEUE_BROWSER_PREFETCH, 1569
 - DEFAULT_QUEUE_PREFETCH, 1569
 - DEFAULT_TOPIC_PREFETCH, 1569
 - DefaultPrefetchPolicy, 1567
 - getDurableTopicPrefetch, 1567
 - getMaxPrefetchLimit, 1567
 - getQueueBrowserPrefetch, 1567
 - getQueuePrefetch, 1568
 - getTopicPrefetch, 1568
 - MAX_PREFETCH_SIZE, 1569
 - setDurableTopicPrefetch, 1568
 - setQueueBrowserPrefetch, 1568
 - setQueuePrefetch, 1568
 - setTopicPrefetch, 1569
- activemq::core::policies::DefaultRedeliveryPolicy, 1569
 - ~DefaultRedeliveryPolicy, 1570
 - clone, 1570
 - DefaultRedeliveryPolicy, 1570
 - getBackOffMultiplier, 1571
 - getCollisionAvoidancePercent, 1571
 - getInitialRedeliveryDelay, 1571
 - getMaximumRedeliveries, 1571
 - getRedeliveryDelay, 1571
 - isUseCollisionAvoidance, 1572
 - isUseExponentialBackOff, 1572
 - setBackOffMultiplier, 1572
 - setCollisionAvoidancePercent, 1572
 - setInitialRedeliveryDelay, 1573
 - setMaximumRedeliveries, 1573
 - setUseCollisionAvoidance, 1573
 - setUseExponentialBackOff, 1573
- activemq::core::PrefetchPolicy, 2783
 - ~PrefetchPolicy, 2785
 - clone, 2785
 - configure, 2785
 - getDurableTopicPrefetch, 2785
 - getMaxPrefetchLimit, 2785
 - getQueueBrowserPrefetch, 2786
 - getQueuePrefetch, 2786
 - getTopicPrefetch, 2786
 - PrefetchPolicy, 2785
 - setDurableTopicPrefetch, 2786
 - setQueueBrowserPrefetch, 2786
 - setQueuePrefetch, 2787
 - setTopicPrefetch, 2787
- activemq::core::RedeliveryPolicy, 2972
 - ~RedeliveryPolicy, 2974
 - clone, 2974
 - configure, 2974
 - getBackOffMultiplier, 2974
 - getCollisionAvoidancePercent, 2975
 - getInitialRedeliveryDelay, 2975
 - getMaximumRedeliveries, 2975
 - getRedeliveryDelay, 2975
 - isUseCollisionAvoidance, 2976
 - isUseExponentialBackOff, 2976
 - NO_MAXIMUM_REDELIVERIES, 2977
 - RedeliveryPolicy, 2974
 - setBackOffMultiplier, 2976
 - setCollisionAvoidancePercent, 2976
 - setInitialRedeliveryDelay, 2976
 - setMaximumRedeliveries, 2976
 - setUseCollisionAvoidance, 2977
 - setUseExponentialBackOff, 2977
- activemq::core::Synchronization, 3477
 - ~Synchronization, 3477
 - afterCommit, 3477
 - afterRollback, 3477
 - beforeEnd, 3477
- activemq::exceptions, 87
- activemq::exceptions::ActiveMQException, 313
 - ~ActiveMQException, 315
 - ActiveMQException, 314
 - clone, 315
 - convertToCMSException, 315
- activemq::exceptions::BrokerException, 797
 - ~BrokerException, 797
 - BrokerException, 797
 - clone, 798
- activemq::io, 87
- activemq::io::LoggingInputStream, 2250
 - ~LoggingInputStream, 2250
 - doReadArrayBounded, 2250
 - doReadByte, 2250
 - LoggingInputStream, 2250
- activemq::io::LoggingOutputStream, 2251
 - ~LoggingOutputStream, 2251
 - doWriteArrayBounded, 2252
 - doWriteByte, 2252
 - LoggingOutputStream, 2251

- activemq::library, 88
- activemq::library::ActiveMQCPP, 277
 - ~ActiveMQCPP, 278
 - ActiveMQCPP, 278
 - initializeLibrary, 278
 - operator=, 279
 - shutdownLibrary, 279
- activemq::state, 88
- activemq::state::CommandVisitor, 1113
 - ~CommandVisitor, 1115
 - processBeginTransaction, 1115
 - processBrokerError, 1115
 - processBrokerInfo, 1115
 - processCommitTransactionOnePhase, 1115
 - processCommitTransactionTwoPhase, 1115
 - processConnectionControl, 1115
 - processConnectionError, 1116
 - processConnectionInfo, 1116
 - processConsumerControl, 1116
 - processConsumerInfo, 1116
 - processControlCommand, 1116
 - processDestinationInfo, 1116
 - processEndTransaction, 1116
 - processFlushCommand, 1116
 - processForgetTransaction, 1117
 - processKeepAliveInfo, 1117
 - processMessage, 1117
 - processMessageAck, 1117
 - processMessageDispatch, 1117
 - processMessageDispatchNotification, 1117
 - processMessagePull, 1117
 - processPrepareTransaction, 1117
 - processProducerAck, 1117
 - processProducerInfo, 1118
 - processRecoverTransactions, 1118
 - processRemoveConnection, 1118
 - processRemoveConsumer, 1118
 - processRemoveDestination, 1118
 - processRemoveInfo, 1118
 - processRemoveProducer, 1118
 - processRemoveSession, 1118
 - processRemoveSubscriptionInfo, 1119
 - processReplayCommand, 1119
 - processResponse, 1119
 - processRollbackTransaction, 1119
 - processSessionInfo, 1119
 - processShutdownInfo, 1119
 - processTransactionInfo, 1119
 - processWireFormat, 1119
- activemq::state::CommandVisitorAdapter, 1120
 - ~CommandVisitorAdapter, 1123
 - processBeginTransaction, 1123
 - processBrokerError, 1123
 - processBrokerInfo, 1123
 - processCommitTransactionOnePhase, 1123
 - processCommitTransactionTwoPhase, 1123
 - processConnectionControl, 1123
 - processConnectionError, 1123
 - processConnectionInfo, 1123
 - processConsumerControl, 1123
 - processConsumerInfo, 1123
 - processControlCommand, 1123
 - processDestinationInfo, 1123
 - processEndTransaction, 1123
 - processFlushCommand, 1123
 - processForgetTransaction, 1123
 - processKeepAliveInfo, 1123
 - processMessage, 1123
 - processMessageAck, 1123
 - processMessageDispatch, 1123
 - processMessageDispatchNotification, 1123
 - processMessagePull, 1123
 - processPrepareTransaction, 1123
 - processProducerAck, 1123
 - processProducerInfo, 1123
 - processRecoverTransactions, 1123
 - processRemoveConnection, 1123
 - processRemoveConsumer, 1123
 - processRemoveDestination, 1123
 - processRemoveInfo, 1123
 - processRemoveProducer, 1124
 - processRemoveSession, 1124
 - processRemoveSubscriptionInfo, 1124
 - processReplayCommand, 1124
 - processResponse, 1124
 - processRollbackTransaction, 1124
 - processSessionInfo, 1124
 - processShutdownInfo, 1124
 - processTransactionInfo, 1124
 - processWireFormat, 1125
- activemq::state::ConnectionState, 1291
 - ~ConnectionState, 1292
 - addSession, 1292
 - addTempDestination, 1292
 - addTransactionState, 1292
 - checkShutdown, 1292
 - ConnectionState, 1292
 - getInfo, 1292
 - getRecoveringPullConsumers, 1292
 - getSessionState, 1292
 - getSessionStates, 1292
 - getTempDestinations, 1292
 - getTransactionState, 1292
 - getTransactionStates, 1292
 - isConnectionInterruptProcessingComplete, 1293

- removeSession, 1293
- removeTempDestination, 1293
- removeTransactionState, 1293
- reset, 1293
- setConnectionInterruptProcessingComplete, 1293
- shutdown, 1293
- toString, 1293
- activemq::state::ConnectionStateTracker, 1293
 - ~ConnectionStateTracker, 1296
 - connectionInterruptProcessingComplete, 1296
 - ConnectionStateTracker, 1296
 - getMaxCacheSize, 1296
 - isRestoreConsumers, 1296
 - isRestoreProducers, 1296
 - isRestoreSessions, 1296
 - isRestoreTransaction, 1296
 - isTrackMessages, 1296
 - isTrackTransactionProducers, 1296
 - isTrackTransactions, 1296
 - processBeginTransaction, 1296
 - processCommitTransactionOnePhase, 1296
 - processCommitTransactionTwoPhase, 1296
 - processConnectionInfo, 1297
 - processConsumerInfo, 1297
 - processDestinationInfo, 1297
 - processEndTransaction, 1297
 - processMessage, 1297
 - processMessageAck, 1297
 - processPrepareTransaction, 1297
 - processProducerInfo, 1298
 - processRemoveConnection, 1298
 - processRemoveConsumer, 1298
 - processRemoveDestination, 1298
 - processRemoveProducer, 1298
 - processRemoveSession, 1298
 - processRollbackTransaction, 1298
 - processSessionInfo, 1299
 - RemoveTransactionAction, 1300
 - restore, 1299
 - setMaxCacheSize, 1300
 - setRestoreConsumers, 1300
 - setRestoreProducers, 1300
 - setRestoreSessions, 1300
 - setRestoreTransaction, 1300
 - setTrackMessages, 1300
 - setTrackTransactionProducers, 1300
 - setTrackTransactions, 1300
 - track, 1300
 - trackBack, 1300
 - transportInterrupted, 1300
- activemq::state::ConsumerState, 1389
 - ~ConsumerState, 1390
 - ConsumerState, 1390
 - getInfo, 1390
 - toString, 1390
- activemq::state::ProducerState, 2926
 - ~ProducerState, 2926
 - getInfo, 2926
 - getTransactionState, 2926
 - ProducerState, 2926
 - setTransactionState, 2926
 - toString, 2926
- activemq::state::SessionState, 3217
 - ~SessionState, 3219
 - addConsumer, 3219
 - addProducer, 3219
 - checkShutdown, 3219
 - getConsumerState, 3219
 - getConsumerStates, 3219
 - getInfo, 3219
 - getProducerState, 3219
 - getProducerStates, 3219
 - removeConsumer, 3219
 - removeProducer, 3219
 - SessionState, 3219
 - shutdown, 3219
 - toString, 3219
- activemq::state::Tracked, 3569
 - ~Tracked, 3569
 - isWaitingForResponse, 3569
 - onResponse, 3569
 - Tracked, 3569
- activemq::state::TransactionState, 3622
 - ~TransactionState, 3624
 - addCommand, 3624
 - addProducerState, 3624
 - checkShutdown, 3624
 - getCommands, 3624
 - getId, 3624
 - getPreparedResult, 3624
 - getProducerStates, 3624
 - isPrepared, 3624
 - setPrepared, 3624
 - setPreparedResult, 3624
 - shutdown, 3624
 - toString, 3624
 - TransactionState, 3624
- activemq::threads, 88
- activemq::threads::CompositeTask, 1132
 - ~CompositeTask, 1132
 - isPending, 1132
- activemq::threads::CompositeTaskRunner, 1133
 - ~CompositeTaskRunner, 1134
 - addTask, 1134
 - CompositeTaskRunner, 1134

- iterate, 1134
- removeTask, 1134
- run, 1134
- shutdown, 1134
- wakeup, 1134
- activemq::threads::DedicatedTaskRunner, 1564
 - ~DedicatedTaskRunner, 1565
 - DedicatedTaskRunner, 1565
 - run, 1565
 - shutdown, 1565
 - wakeup, 1565
- activemq::threads::Task, 3494
 - ~Task, 3495
 - iterate, 3495
- activemq::threads::TaskRunner, 3496
 - ~TaskRunner, 3497
 - shutdown, 3497
 - wakeup, 3497
- activemq::transport, 89
- activemq::transport::AbstractTransportFactory, 164
 - ~AbstractTransportFactory, 165
 - createWireFormat, 165
- activemq::transport::CompositeTransport, 1135
 - ~CompositeTransport, 1136
 - addURI, 1136
 - removeURI, 1136
- activemq::transport::correlator, 89
- activemq::transport::correlator::FutureResponse, 1843
 - ~FutureResponse, 1844
 - FutureResponse, 1844
 - getResponse, 1844
 - setResponse, 1844
- activemq::transport::correlator::ResponseCorrelator, 3081
 - ~ResponseCorrelator, 3082
 - close, 3082
 - onCommand, 3082
 - oneway, 3083
 - onTransportException, 3083
 - request, 3083, 3084
 - ResponseCorrelator, 3082
 - start, 3084
- activemq::transport::DefaultTransportListener, 1593
 - ~DefaultTransportListener, 1594
 - onCommand, 1594
 - onException, 1594
 - transportInterrupted, 1594
 - transportResumed, 1594
- activemq::transport::failover, 90
- activemq::transport::failover::BackupTransport, 690
 - ~BackupTransport, 690
 - BackupTransport, 690
 - getTransport, 690
 - getUri, 691
 - isClosed, 691
 - onException, 691
 - setClosed, 691
 - setTransport, 691
 - setUri, 691
- activemq::transport::failover::BackupTransportPool, 692
 - ~BackupTransportPool, 693
 - BackupTransport, 694
 - BackupTransportPool, 693
 - getBackup, 693
 - getBackupPoolSize, 693
 - isEnabled, 693
 - isPending, 693
 - iterate, 694
 - setBackupPoolSize, 694
 - setEnabled, 694
- activemq::transport::failover::CloseTransportsTask, 1066
 - ~CloseTransportsTask, 1067
 - add, 1067
 - CloseTransportsTask, 1067
 - isPending, 1067
 - iterate, 1067
- activemq::transport::failover::FailoverTransport, 1752
 - ~FailoverTransport, 1755
 - add, 1755
 - addURI, 1755
 - close, 1755
 - FailoverTransport, 1755
 - FailoverTransportListener, 1763
 - getBackOffMultiplier, 1756
 - getBackupPoolSize, 1756
 - getInitialReconnectDelay, 1756
 - getMaxCacheSize, 1756
 - getMaxReconnectAttempts, 1756
 - getMaxReconnectDelay, 1756
 - getReconnectDelay, 1756
 - getRemoteAddress, 1756
 - getStartupMaxReconnectAttempts, 1756
 - getTimeout, 1756
 - getTransportListener, 1756
 - handleTransportFailure, 1757
 - isBackup, 1757
 - isClosed, 1757
 - isConnected, 1757
 - isFaultTolerant, 1757
 - isInitialized, 1757
 - isPending, 1758

- isRandomize, 1758
- isTrackMessages, 1758
- isTrackTransactionProducers, 1758
- isUseExponentialBackOff, 1758
- iterate, 1758
- narrow, 1758
- oneway, 1759
- reconnect, 1759
- removeURI, 1759
- request, 1760
- restoreTransport, 1760
- setBackOffMultiplier, 1761
- setBackup, 1761
- setBackupPoolSize, 1761
- setConnectionInterruptProcessingComplete, 1761
- setInitialized, 1761
- setInitialReconnectDelay, 1761
- setMaxCacheSize, 1762
- setMaxReconnectAttempts, 1762
- setMaxReconnectDelay, 1762
- setRandomize, 1762
- setReconnectDelay, 1762
- setStartupMaxReconnectAttempts, 1762
- setTimeout, 1762
- setTrackMessages, 1762
- setTrackTransactionProducers, 1762
- setTransportListener, 1762
- setUseExponentialBackOff, 1762
- setWireFormat, 1763
- start, 1763
- stop, 1763
- activemq::transport::failover::FailoverTransportFactory, 1763
 - ~FailoverTransportFactory, 1764
 - create, 1764
 - createComposite, 1765
 - doCreateComposite, 1765
- activemq::transport::failover::FailoverTransportListener, 1766
 - ~FailoverTransportListener, 1767
 - FailoverTransportListener, 1767
 - onCommand, 1767
 - onException, 1767
 - transportInterrupted, 1767
 - transportResumed, 1767
- activemq::transport::failover::URIPool, 3681
 - ~URIPool, 3682
 - addURI, 3682
 - addURIs, 3683
 - getURI, 3683
 - isRandomize, 3683
 - removeURI, 3683
 - setRandomize, 3683
 - URIPool, 3682
- activemq::transport::inactivity, 90
- activemq::transport::inactivity::InactivityMonitor, 1874
 - ~InactivityMonitor, 1875
 - AsyncSignalReadErrorTask, 1877
 - AsyncWriteTask, 1877
 - close, 1875
 - getInitialDelayTime, 1875
 - getReadCheckTime, 1876
 - getWriteCheckTime, 1876
 - InactivityMonitor, 1875
 - isKeepAliveResponseRequired, 1876
 - onCommand, 1876
 - oneway, 1876
 - onException, 1876
 - ReadChecker, 1877
 - setInitialDelayTime, 1877
 - setKeepAliveResponseRequired, 1877
 - setReadCheckTime, 1877
 - setWriteCheckTime, 1877
 - WriteChecker, 1877
- activemq::transport::inactivity::ReadChecker, 2959
 - ~ReadChecker, 2960
 - ReadChecker, 2960
 - run, 2960
- activemq::transport::inactivity::WriteChecker, 3755
 - ~WriteChecker, 3755
 - run, 3755
 - WriteChecker, 3755
- activemq::transport::IOTransport, 2005
 - ~IOTransport, 2007
 - close, 2007
 - getRemoteAddress, 2008
 - getTransportListener, 2008
 - IOTransport, 2007
 - isClosed, 2008
 - isConnected, 2008
 - isFaultTolerant, 2008
 - narrow, 2008
 - oneway, 2009
 - reconnect, 2009
 - request, 2009, 2010
 - run, 2010
 - setInputStream, 2010
 - setOutputStream, 2010
 - setTransportListener, 2010
 - setWireFormat, 2011
 - start, 2011
 - stop, 2011
- activemq::transport::logging, 90

- activemq::transport::logging::LoggingTransport, 2252
 - ~LoggingTransport, 2253
 - LoggingTransport, 2253
 - onCommand, 2253
 - oneway, 2253
 - request, 2254
- activemq::transport::mock, 90
- activemq::transport::mock::InternalCommandListener, start, 2601
 - 1986
 - ~InternalCommandListener, 1987
 - InternalCommandListener, 1987
 - onCommand, 1987
 - run, 1987
 - setResponseBuilder, 1987
 - setTransport, 1987
- activemq::transport::mock::MockTransport, 2592
 - ~MockTransport, 2594
 - close, 2594
 - fireCommand, 2595
 - fireException, 2595
 - getInstance, 2595
 - getNumReceivedMessageBeforeFail, 2596
 - getNumReceivedMessages, 2596
 - getNumSentKeepAlives, 2596
 - getNumSentKeepAlivesBeforeFail, 2596
 - getNumSentMessageBeforeFail, 2596
 - getNumSentMessages, 2596
 - getRemoteAddress, 2596
 - getTransportListener, 2596
 - getWireFormat, 2596
 - isClosed, 2597
 - isConnected, 2597
 - isFailOnClose, 2597
 - isFailOnKeepAliveSends, 2597
 - isFailOnReceiveMessage, 2597
 - isFailOnSendMessage, 2597
 - isFailOnStart, 2597
 - isFailOnStop, 2597
 - isFaultTolerant, 2597
 - MockTransport, 2594
 - narrow, 2598
 - oneway, 2598
 - reconnect, 2598
 - request, 2598, 2599
 - setFailOnClose, 2599
 - setFailOnKeepAliveSends, 2600
 - setFailOnReceiveMessage, 2600
 - setFailOnSendMessage, 2600
 - setFailOnStart, 2600
 - setFailOnStop, 2600
 - setNumReceivedMessageBeforeFail, 2600
 - setNumReceivedMessages, 2600
 - setNumSentKeepAlives, 2600
 - setNumSentKeepAlivesBeforeFail, 2600
 - setNumSentMessageBeforeFail, 2600
 - setNumSentMessages, 2600
 - setOutgoingListener, 2600
 - setResponseBuilder, 2600
 - setTransportListener, 2601
 - setWireFormat, 2601
- activemq::transport::mock::MockTransportFactory, 2602
 - ~MockTransportFactory, 2603
 - create, 2603
 - createComposite, 2603
 - doCreateComposite, 2603
- activemq::transport::mock::ResponseBuilder, 3079
 - ~ResponseBuilder, 3080
 - buildIncomingCommands, 3080
 - buildResponse, 3080
- activemq::transport::tcp, 91
- activemq::transport::tcp::SslTransport, 3347
 - ~SslTransport, 3348
 - configureSocket, 3349
 - createSocket, 3349
 - SslTransport, 3348
- activemq::transport::tcp::SslTransportFactory, 3349
 - ~SslTransportFactory, 3350
 - doCreateComposite, 3350
- activemq::transport::tcp::TcpTransport, 3510
 - ~TcpTransport, 3511
 - close, 3512
 - configureSocket, 3512
 - connect, 3512
 - createSocket, 3512
 - isClosed, 3513
 - isConnected, 3513
 - isFaultTolerant, 3513
 - TcpTransport, 3511
- activemq::transport::tcp::TcpTransportFactory, 3514
 - ~TcpTransportFactory, 3515
 - create, 3515
 - createComposite, 3515
 - doCreateComposite, 3515
- activemq::transport::Transport, 3629
 - ~Transport, 3630
 - getRemoteAddress, 3630
 - getTransportListener, 3630
 - isClosed, 3631
 - isConnected, 3631
 - isFaultTolerant, 3631

- narrow, 3631
- oneway, 3632
- reconnect, 3632
- request, 3632, 3633
- setTransportListener, 3633
- setWireFormat, 3633
- start, 3633
- stop, 3634
- activemq::transport::TransportFactory, 3634
 - ~TransportFactory, 3635
 - create, 3635
 - createComposite, 3635
- activemq::transport::TransportFilter, 3636
 - ~TransportFilter, 3638
 - close, 3638
 - fire, 3638
 - getRemoteAddress, 3639
 - getTransportListener, 3639
 - isClosed, 3639
 - isConnected, 3639
 - isFaultTolerant, 3639
 - listener, 3643
 - narrow, 3640
 - next, 3643
 - onCommand, 3640
 - oneway, 3640
 - onException, 3641
 - reconnect, 3641
 - request, 3641, 3642
 - setTransportListener, 3642
 - setWireFormat, 3642
 - start, 3642
 - stop, 3643
 - TransportFilter, 3638
 - transportInterrupted, 3643
 - transportResumed, 3643
- activemq::transport::TransportListener, 3643
 - ~TransportListener, 3644
 - onCommand, 3644
 - onException, 3644
 - transportInterrupted, 3645
 - transportResumed, 3645
- activemq::transport::TransportRegistry, 3645
 - ~TransportRegistry, 3646
 - findFactory, 3646
 - getInstance, 3647
 - getTransportNames, 3647
 - registerFactory, 3647
 - unregisterFactory, 3647
- activemq::util, 91
- activemq::util::ActiveMQProperties, 431
 - ~ActiveMQProperties, 432
 - ActiveMQProperties, 432
 - clear, 432
 - clone, 432
 - copy, 432
 - getProperties, 432, 433
 - getProperty, 433
 - hasProperty, 433
 - isEmpty, 433
 - remove, 434
 - setProperties, 434
 - setProperty, 434
 - toArray, 434
 - toString, 434
- activemq::util::CMSExceptionSupport, 1077
 - ~CMSExceptionSupport, 1078
 - create, 1078
 - createMessageEOFException, 1078
 - createMessageFormatException, 1078
- activemq::util::CompositeData, 1129
 - ~CompositeData, 1131
 - CompositeData, 1131
 - getComponents, 1131
 - getFragment, 1131
 - getHost, 1131
 - getParameters, 1131
 - getPath, 1131
 - getScheme, 1131
 - setComponents, 1131
 - setFragment, 1131
 - setHost, 1131
 - setParameters, 1131
 - setPath, 1131
 - setScheme, 1131
 - toURI, 1131
- activemq::util::IdGenerator, 1861
 - ~IdGenerator, 1862
 - compare, 1862
 - generateId, 1862
 - getHostname, 1862
 - getSeedFromId, 1862
 - getSequenceFromId, 1862
 - IdGenerator, 1862
- activemq::util::LongSequenceGenerator, 2302
 - ~LongSequenceGenerator, 2302
 - getLastSequenceId, 2302
 - getNextSequenceId, 2302
 - LongSequenceGenerator, 2302
- activemq::util::MarshallingSupport, 2335
 - ~MarshallingSupport, 2336
 - asciiToModifiedUtf8, 2336
 - MarshallingSupport, 2336
 - modifiedUtf8ToAscii, 2336
 - readString16, 2337
 - readString32, 2337
 - writeString, 2338
 - writeString16, 2338

- writeString32, 2338
- activemq::util::MemoryUsage, 2354
 - ~MemoryUsage, 2356
 - decreaseUsage, 2356
 - enqueueUsage, 2356
 - getLimit, 2356
 - getUsage, 2356
 - increaseUsage, 2356
 - isFull, 2357
 - MemoryUsage, 2355
 - setLimit, 2357
 - setUsage, 2357
 - waitForSpace, 2357
- activemq::util::PrimitiveList, 2787
 - ~PrimitiveList, 2790
 - getBool, 2791
 - getByte, 2791
 - getByteArray, 2791
 - getChar, 2792
 - getDouble, 2792
 - getFloat, 2792
 - getInt, 2793
 - getLong, 2793
 - getShort, 2794
 - getString, 2794
 - PrimitiveList, 2790
 - setBool, 2794
 - setByte, 2795
 - setByteArray, 2795
 - setChar, 2795
 - setDouble, 2796
 - setFloat, 2796
 - setInt, 2796
 - setLong, 2797
 - setShort, 2797
 - setString, 2797
 - toString, 2798
- activemq::util::PrimitiveMap, 2798
 - ~PrimitiveMap, 2800
 - getBool, 2801
 - getByte, 2801
 - getByteArray, 2802
 - getChar, 2802
 - getDouble, 2802
 - getFloat, 2803
 - getInt, 2803
 - getLong, 2804
 - getShort, 2804
 - getString, 2804
 - PrimitiveMap, 2800, 2801
 - setBool, 2805
 - setByte, 2805
 - setByteArray, 2805
 - setChar, 2806
 - setDouble, 2806
 - setFloat, 2806
 - setInt, 2806
 - setLong, 2806
 - setShort, 2807
 - setString, 2807
 - toString, 2807
- activemq::util::PrimitiveValueConverter, 2816
 - ~PrimitiveValueConverter, 2817
 - convert, 2817
 - PrimitiveValueConverter, 2817
- activemq::util::PrimitiveValueNode, 2817
 - ~PrimitiveValueNode, 2824
 - BIG_STRING_TYPE, 2821
 - BOOLEAN_TYPE, 2821
 - BYTE_ARRAY_TYPE, 2821
 - BYTE_TYPE, 2821
 - CHAR_TYPE, 2821
 - clear, 2824
 - DOUBLE_TYPE, 2821
 - FLOAT_TYPE, 2821
 - getBool, 2824
 - getByte, 2824
 - getByteArray, 2824
 - getChar, 2825
 - getDouble, 2825
 - getFloat, 2825
 - getInt, 2825
 - getList, 2826
 - getLong, 2826
 - getMap, 2826
 - getShort, 2826
 - getString, 2827
 - getType, 2827
 - getValue, 2827
 - INTEGER_TYPE, 2821
 - LIST_TYPE, 2821
 - LONG_TYPE, 2821
 - MAP_TYPE, 2821
 - NULL_TYPE, 2821
 - operator=, 2827
 - operator==, 2827
 - PrimitiveType, 2821
 - PrimitiveValueNode, 2821–2823
 - setBool, 2828
 - setByte, 2828
 - setByteArray, 2828
 - setChar, 2828
 - setDouble, 2828
 - setFloat, 2829
 - setInt, 2829
 - setList, 2829
 - setLong, 2829
 - setMap, 2829

- setShort, 2830
- setString, 2830
- setValue, 2830
- SHORT_TYPE, 2821
- STRING_TYPE, 2821
- toString, 2830
- activemq::util::PrimitiveValueNode::PrimitiveValue, 2814
- boolValue, 2815
- byteArrayValue, 2815
- byteValue, 2815
- charValue, 2815
- doubleValue, 2815
- floatValue, 2815
- intValue, 2815
- listValue, 2815
- longValue, 2815
- mapValue, 2815
- shortValue, 2815
- stringValue, 2815
- activemq::util::URISupport, 3684
- createQueryString, 3685
- parseComposite, 3685
- parseQuery, 3685
- parseURL, 3686
- activemq::util::Usage, 3701
- ~Usage, 3702
- decreaseUsage, 3702
- enqueueUsage, 3702
- increaseUsage, 3702
- isFull, 3702
- waitForSpace, 3703
- activemq::wireformat, 92
- activemq::wireformat::MarshalAware, 2328
- ~MarshalAware, 2329
- afterMarshal, 2329
- afterUnmarshal, 2329
- beforeMarshal, 2329
- beforeUnmarshal, 2330
- getMarshaledForm, 2330
- isMarshalAware, 2330
- setMarshaledForm, 2330
- activemq::wireformat::openwire, 92
- activemq::wireformat::openwire::marshal, 93
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 741
- ~BaseDataStreamMarshaller, 748
- looseMarshal, 748
- looseMarshalBrokerError, 748
- looseMarshalCachedObject, 748
- looseMarshalLong, 749
- looseMarshalNestedObject, 749
- looseMarshalObjectArray, 749
- looseMarshalString, 750
- looseUnmarshal, 750
- looseUnmarshalBrokerError, 751
- looseUnmarshalByteArray, 751
- looseUnmarshalCachedObject, 751
- looseUnmarshalConstByteArray, 752
- looseUnmarshalLong, 752
- looseUnmarshalNestedObject, 752
- looseUnmarshalString, 753
- readAsciiString, 753
- tightMarshal1, 753
- tightMarshal2, 754
- tightMarshalBrokerError1, 754
- tightMarshalBrokerError2, 755
- tightMarshalCachedObject1, 755
- tightMarshalCachedObject2, 755
- tightMarshalLong1, 756
- tightMarshalLong2, 756
- tightMarshalNestedObject1, 757
- tightMarshalNestedObject2, 757
- tightMarshalObjectArray1, 757
- tightMarshalObjectArray2, 758
- tightMarshalString1, 758
- tightMarshalString2, 759
- tightUnmarshal, 759
- tightUnmarshalBrokerError, 760
- tightUnmarshalByteArray, 760
- tightUnmarshalCachedObject, 760
- tightUnmarshalConstByteArray, 761
- tightUnmarshalLong, 761
- tightUnmarshalNestedObject, 762
- tightUnmarshalString, 762
- toHexFromBytes, 762
- toString, 763
- activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1503
- ~DataStreamMarshaller, 1505
- createObject, 1505
- getDataStructureType, 1511
- looseMarshal, 1517
- looseUnmarshal, 1525
- tightMarshal1, 1532
- tightMarshal2, 1539
- tightUnmarshal, 1546
- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2808
- ~PrimitiveTypesMarshaller, 2810
- marshal, 2810
- marshalList, 2810
- marshalMap, 2811
- marshalPrimitive, 2811
- marshalPrimitiveList, 2811
- marshalPrimitiveMap, 2812
- PrimitiveTypesMarshaller, 2810
- unmarshal, 2812

- unmarshalList, 2813
- unmarshalMap, 2813
- unmarshalPrimitive, 2813
- unmarshalPrimitiveList, 2814
- unmarshalPrimitiveMap, 2814
- activemq::wireformat::openwire::marshal::v1, 93
- activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 174
 - ~ActiveMQBlobMessageMarshaller, 176
 - ActiveMQBlobMessageMarshaller, 176
 - createObject, 176
 - getDataStructureType, 176
 - looseMarshal, 176
 - looseUnmarshal, 176
 - tightMarshal1, 177
 - tightMarshal2, 177
 - tightUnmarshal, 178
- activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 213
 - ~ActiveMQBytesMessageMarshaller, 215
 - ActiveMQBytesMessageMarshaller, 215
 - createObject, 215
 - getDataStructureType, 215
 - looseMarshal, 215
 - looseUnmarshal, 215
 - tightMarshal1, 216
 - tightMarshal2, 216
 - tightUnmarshal, 217
- activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 293
 - ~ActiveMQDestinationMarshaller, 295
 - ActiveMQDestinationMarshaller, 295
 - looseMarshal, 295
 - looseUnmarshal, 295
 - tightMarshal1, 296
 - tightMarshal2, 296
 - tightUnmarshal, 297
- activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 332
 - ~ActiveMQMapMessageMarshaller, 334
 - ActiveMQMapMessageMarshaller, 334
 - createObject, 334
 - getDataStructureType, 334
 - looseMarshal, 334
 - looseUnmarshal, 334
 - tightMarshal1, 335
 - tightMarshal2, 335
 - tightUnmarshal, 336
- activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 359
 - ~ActiveMQMessageMarshaller, 360
 - ActiveMQMessageMarshaller, 360
 - createObject, 360
 - getDataStructureType, 360
 - looseMarshal, 360
 - looseUnmarshal, 361
 - tightMarshal1, 361
 - tightMarshal2, 362
 - tightUnmarshal, 362
- activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 403
 - ~ActiveMQObjectMessageMarshaller, 404
 - ActiveMQObjectMessageMarshaller, 404
 - createObject, 404
 - getDataStructureType, 404
 - looseMarshal, 404
 - looseUnmarshal, 405
 - tightMarshal1, 405
 - tightMarshal2, 406
 - tightUnmarshal, 406
- activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMessageMarshaller, 445
 - ~ActiveMQQueueMarshaller, 446
 - ActiveMQQueueMarshaller, 446
 - createObject, 446
 - getDataStructureType, 446
 - looseMarshal, 446
 - looseUnmarshal, 447
 - tightMarshal1, 447
 - tightMarshal2, 448
 - tightUnmarshal, 448
- activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 503
 - ~ActiveMQStreamMessageMarshaller, 505
 - ActiveMQStreamMessageMarshaller, 505
 - createObject, 505
 - getDataStructureType, 505
 - looseMarshal, 505
 - looseUnmarshal, 505
 - tightMarshal1, 506
 - tightMarshal2, 506
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 530
 - ~ActiveMQTempDestinationMarshaller, 531
 - ActiveMQTempDestinationMarshaller, 531
 - looseMarshal, 531
 - looseUnmarshal, 532
 - tightMarshal1, 532
 - tightMarshal2, 533
 - tightUnmarshal, 533
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 556
 - ~ActiveMQTempQueueMarshaller, 558
 - ActiveMQTempQueueMarshaller, 558
 - createObject, 558

- getDataStructureType, 558
 - looseMarshal, 558
 - looseUnmarshal, 558
 - tightMarshal1, 559
 - tightMarshal2, 559
 - tightUnmarshal, 560
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 588
 - ~ActiveMQTempTopicMarshaller, 590
 - ActiveMQTempTopicMarshaller, 590
 - createObject, 590
 - getDataStructureType, 590
 - looseMarshal, 590
 - looseUnmarshal, 590
 - tightMarshal1, 591
 - tightMarshal2, 591
 - tightUnmarshal, 592
- activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 617
 - ~ActiveMQTextMessageMarshaller, 618
 - ActiveMQTextMessageMarshaller, 618
 - createObject, 618
 - getDataStructureType, 618
 - looseMarshal, 618
 - looseUnmarshal, 619
 - tightMarshal1, 619
 - tightMarshal2, 620
 - tightUnmarshal, 620
- activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 644
 - ~ActiveMQTopicMarshaller, 646
 - ActiveMQTopicMarshaller, 646
 - createObject, 646
 - getDataStructureType, 646
 - looseMarshal, 646
 - looseUnmarshal, 646
 - tightMarshal1, 647
 - tightMarshal2, 647
 - tightUnmarshal, 648
- activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 714
 - ~BaseCommandMarshaller, 716
 - BaseCommandMarshaller, 716
 - looseMarshal, 716
 - looseUnmarshal, 717
 - tightMarshal1, 718
 - tightMarshal2, 719
 - tightUnmarshal, 720
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 808
 - ~BrokerIdMarshaller, 810
 - BrokerIdMarshaller, 810
 - createObject, 810
 - getDataStructureType, 810
 - looseMarshal, 810
 - looseUnmarshal, 810
 - tightMarshal1, 811
 - tightMarshal2, 811
 - tightUnmarshal, 812
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 839
 - ~BrokerInfoMarshaller, 841
 - BrokerInfoMarshaller, 841
 - createObject, 841
 - getDataStructureType, 841
 - looseMarshal, 841
 - looseUnmarshal, 841
 - tightMarshal1, 842
 - tightMarshal2, 842
 - tightUnmarshal, 843
- activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1184
 - ~ConnectionControlMarshaller, 1186
 - ConnectionControlMarshaller, 1186
 - createObject, 1186
 - getDataStructureType, 1186
 - looseMarshal, 1186
 - looseUnmarshal, 1186
 - tightMarshal1, 1187
 - tightMarshal2, 1187
 - tightUnmarshal, 1188
- activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1216
 - ~ConnectionErrorMarshaller, 1217
 - ConnectionErrorMarshaller, 1217
 - createObject, 1217
 - getDataStructureType, 1217
 - looseMarshal, 1217
 - looseUnmarshal, 1218
 - tightMarshal1, 1218
 - tightMarshal2, 1219
 - tightUnmarshal, 1219
- activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1247
 - ~ConnectionIdMarshaller, 1247
 - ConnectionIdMarshaller, 1247
 - createObject, 1247
 - getDataStructureType, 1247
 - looseMarshal, 1247
 - looseUnmarshal, 1247
 - tightMarshal1, 1248
 - tightMarshal2, 1248
 - tightUnmarshal, 1249
- activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1275
 - ~ConnectionInfoMarshaller, 1277
 - ConnectionInfoMarshaller, 1277
 - createObject, 1277

- getDataStructureType, 1277
 - looseMarshal, 1277
 - looseUnmarshal, 1277
 - tightMarshal1, 1278
 - tightMarshal2, 1278
 - tightUnmarshal, 1279
- activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1318
 - ~ConsumerControlMarshaller, 1320
 - ConsumerControlMarshaller, 1320
 - createObject, 1320
 - getDataStructureType, 1320
 - looseMarshal, 1320
 - looseUnmarshal, 1320
 - tightMarshal1, 1321
 - tightMarshal2, 1321
 - tightUnmarshal, 1322
- activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1345
 - ~ConsumerIdMarshaller, 1347
 - ConsumerIdMarshaller, 1347
 - createObject, 1347
 - getDataStructureType, 1347
 - looseMarshal, 1347
 - looseUnmarshal, 1347
 - tightMarshal1, 1348
 - tightMarshal2, 1348
 - tightUnmarshal, 1349
- activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1377
 - ~ConsumerInfoMarshaller, 1379
 - ConsumerInfoMarshaller, 1379
 - createObject, 1379
 - getDataStructureType, 1379
 - looseMarshal, 1379
 - looseUnmarshal, 1379
 - tightMarshal1, 1380
 - tightMarshal2, 1380
 - tightUnmarshal, 1381
- activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1405
 - ~ControlCommandMarshaller, 1406
 - ControlCommandMarshaller, 1406
 - createObject, 1406
 - getDataStructureType, 1406
 - looseMarshal, 1406
 - looseUnmarshal, 1407
 - tightMarshal1, 1407
 - tightMarshal2, 1408
 - tightUnmarshal, 1408
- activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1438
 - ~DataArrayResponseMarshaller, 1439
 - createObject, 1439
- DataArrayResponseMarshaller, 1439
 - getDataStructureType, 1439
 - looseMarshal, 1439
 - looseUnmarshal, 1440
 - tightMarshal1, 1440
 - tightMarshal2, 1441
- activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1499
 - ~DataResponseMarshaller, 1501
 - createObject, 1501
 - DataResponseMarshaller, 1501
 - getDataStructureType, 1501
 - looseMarshal, 1501
 - looseUnmarshal, 1502
 - tightMarshal1, 1502
 - tightMarshal2, 1502
- activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1630
 - ~DestinationInfoMarshaller, 1631
 - createObject, 1631
 - DestinationInfoMarshaller, 1631
 - getDataStructureType, 1631
 - looseMarshal, 1631
 - looseUnmarshal, 1632
 - tightMarshal1, 1632
 - tightMarshal2, 1633
- activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1662
 - ~DiscoveryEventMarshaller, 1664
 - createObject, 1664
 - DiscoveryEventMarshaller, 1664
 - getDataStructureType, 1664
 - looseMarshal, 1664
 - looseUnmarshal, 1664
 - tightMarshal1, 1665
 - tightMarshal2, 1665
- activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1742
 - ~ExceptionResponseMarshaller, 1744
 - createObject, 1744
 - ExceptionResponseMarshaller, 1744
 - getDataStructureType, 1744
 - looseMarshal, 1744
 - looseUnmarshal, 1745
 - tightMarshal1, 1745
 - tightMarshal2, 1745
- activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1830
 - ~FlushCommandMarshaller, 1832

- createObject, 1832
- FlushCommandMarshaller, 1832
- getDataStructureType, 1832
- looseMarshal, 1832
- looseUnmarshal, 1832
- tightMarshal1, 1833
- tightMarshal2, 1833
- tightUnmarshal, 1834
- activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 1974
 - ~IntegerResponseMarshaller, 1976
 - createObject, 1976
 - getDataStructureType, 1976
 - IntegerResponseMarshaller, 1976
 - looseMarshal, 1976
 - looseUnmarshal, 1977
 - tightMarshal1, 1977
 - tightMarshal2, 1977
 - tightUnmarshal, 1978
- activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2036
 - ~JournalQueueAckMarshaller, 2038
 - createObject, 2038
 - getDataStructureType, 2038
 - JournalQueueAckMarshaller, 2038
 - looseMarshal, 2038
 - looseUnmarshal, 2038
 - tightMarshal1, 2039
 - tightMarshal2, 2039
 - tightUnmarshal, 2040
- activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2065
 - ~JournalTopicAckMarshaller, 2067
 - createObject, 2067
 - getDataStructureType, 2067
 - JournalTopicAckMarshaller, 2067
 - looseMarshal, 2067
 - looseUnmarshal, 2067
 - tightMarshal1, 2068
 - tightMarshal2, 2068
 - tightUnmarshal, 2069
- activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2087
 - ~JournalTraceMarshaller, 2089
 - createObject, 2089
 - getDataStructureType, 2089
 - JournalTraceMarshaller, 2089
 - looseMarshal, 2089
 - looseUnmarshal, 2089
 - tightMarshal1, 2090
 - tightMarshal2, 2090
 - tightUnmarshal, 2091
- activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2118
 - ~JournalTransactionMarshaller, 2120
 - createObject, 2120
 - getDataStructureType, 2120
 - JournalTransactionMarshaller, 2120
 - looseMarshal, 2120
 - looseUnmarshal, 2120
 - tightMarshal1, 2121
 - tightMarshal2, 2121
- activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2145
 - ~KeepAliveInfoMarshaller, 2146
 - createObject, 2146
 - getDataStructureType, 2146
 - KeepAliveInfoMarshaller, 2146
 - looseMarshal, 2146
 - looseUnmarshal, 2147
 - tightMarshal1, 2147
 - tightMarshal2, 2148
- activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2178
 - ~LastPartialCommandMarshaller, 2179
 - createObject, 2179
 - getDataStructureType, 2179
 - LastPartialCommandMarshaller, 2179
 - looseMarshal, 2180
 - looseUnmarshal, 2180
 - tightMarshal1, 2180
 - tightMarshal2, 2181
- activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2224
 - ~LocalTransactionIdMarshaller, 2225
 - createObject, 2225
 - getDataStructureType, 2225
 - LocalTransactionIdMarshaller, 2225
 - looseMarshal, 2225
 - looseUnmarshal, 2226
 - tightMarshal1, 2226
 - tightMarshal2, 2227
- activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 2334
 - ~MarshallerFactory, 2334
 - configure, 2334
- activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2415
 - ~MessageAckMarshaller, 2416
 - createObject, 2416
 - getDataStructureType, 2416
 - looseMarshal, 2416
 - looseUnmarshal, 2417
 - tightMarshal1, 2417
 - MessageAckMarshaller, 2416

- tightMarshal1, 2417
- tightMarshal2, 2418
- tightUnmarshal, 2418
- activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2454
 - ~MessageDispatchMarshaller, 2455
 - createObject, 2455
 - getDataStructureType, 2455
 - looseMarshal, 2455
 - looseUnmarshal, 2456
 - MessageDispatchMarshaller, 2455
 - tightMarshal1, 2456
 - tightMarshal2, 2457
 - tightUnmarshal, 2457
- activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2482
 - ~MessageDispatchNotificationMarshaller, 2484
 - createObject, 2484
 - getDataStructureType, 2484
 - looseMarshal, 2484
 - looseUnmarshal, 2484
 - MessageDispatchNotificationMarshaller, 2484
 - tightMarshal1, 2485
 - tightMarshal2, 2485
 - tightUnmarshal, 2486
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2518
 - ~MessageIdMarshaller, 2520
 - createObject, 2520
 - getDataStructureType, 2520
 - looseMarshal, 2520
 - looseUnmarshal, 2520
 - MessageIdMarshaller, 2520
 - tightMarshal1, 2521
 - tightMarshal2, 2521
 - tightUnmarshal, 2522
- activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2540
 - ~MessageMarshaller, 2541
 - looseMarshal, 2541
 - looseUnmarshal, 2541
 - MessageMarshaller, 2541
 - tightMarshal1, 2542
 - tightMarshal2, 2542
 - tightUnmarshal, 2543
- activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2584
 - ~MessagePullMarshaller, 2585
 - createObject, 2585
 - getDataStructureType, 2585
 - looseMarshal, 2585
 - looseUnmarshal, 2586
- MessagePullMarshaller, 2585
- tightMarshal1, 2586
- tightMarshal2, 2587
- activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2636
 - ~NetworkBridgeFilterMarshaller, 2638
 - createObject, 2638
 - getDataStructureType, 2638
 - looseMarshal, 2638
 - looseUnmarshal, 2638
 - NetworkBridgeFilterMarshaller, 2638
 - tightMarshal1, 2639
 - tightMarshal2, 2639
- activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2752
 - ~PartialCommandMarshaller, 2753
 - createObject, 2753
 - getDataStructureType, 2753
 - looseMarshal, 2754
 - looseUnmarshal, 2754
 - PartialCommandMarshaller, 2753
 - tightMarshal1, 2754
 - tightMarshal2, 2755
 - tightUnmarshal, 2755
- activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 2862
 - ~ProducerAckMarshaller, 2864
 - createObject, 2864
 - getDataStructureType, 2864
 - looseMarshal, 2864
 - looseUnmarshal, 2864
 - ProducerAckMarshaller, 2864
 - tightMarshal1, 2865
 - tightMarshal2, 2865
 - tightUnmarshal, 2866
- activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 2895
 - ~ProducerIdMarshaller, 2895
 - createObject, 2895
 - getDataStructureType, 2895
 - looseMarshal, 2895
 - looseUnmarshal, 2895
 - ProducerIdMarshaller, 2895
 - tightMarshal1, 2896
 - tightMarshal2, 2896
- activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 2910
 - ~ProducerInfoMarshaller, 2911
 - createObject, 2911
 - getDataStructureType, 2911
 - looseMarshal, 2911

- looseUnmarshal, 2912
 - ProducerInfoMarshaller, 2911
 - tightMarshal1, 2912
 - tightMarshal2, 2913
 - tightUnmarshal, 2913
- activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 3003
 - ~RemoveInfoMarshaller, 3004
 - createObject, 3004
 - getDataStructureType, 3004
 - looseMarshal, 3004
 - looseUnmarshal, 3005
 - RemoveInfoMarshaller, 3004
 - tightMarshal1, 3005
 - tightMarshal2, 3006
 - tightUnmarshal, 3006
- activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3019
 - ~RemoveSubscriptionInfoMarshaller, 3020
 - createObject, 3020
 - getDataStructureType, 3020
 - looseMarshal, 3021
 - looseUnmarshal, 3021
 - RemoveSubscriptionInfoMarshaller, 3020
 - tightMarshal1, 3021
 - tightMarshal2, 3022
 - tightUnmarshal, 3022
- activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 3050
 - ~ReplayCommandMarshaller, 3051
 - createObject, 3051
 - getDataStructureType, 3051
 - looseMarshal, 3051
 - looseUnmarshal, 3052
 - ReplayCommandMarshaller, 3051
 - tightMarshal1, 3052
 - tightMarshal2, 3053
 - tightUnmarshal, 3053
- activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 3102
 - ~ResponseMarshaller, 3104
 - createObject, 3104
 - getDataStructureType, 3104
 - looseMarshal, 3104
 - looseUnmarshal, 3105
 - ResponseMarshaller, 3104
 - tightMarshal1, 3105
 - tightMarshal2, 3106
 - tightUnmarshal, 3106
- activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 3184
 - ~SessionIdMarshaller, 3186
 - createObject, 3186
 - getDataStructureType, 3186
- looseMarshal, 3186
 - looseUnmarshal, 3186
 - SessionIdMarshaller, 3186
 - tightMarshal1, 3187
 - tightMarshal2, 3187
- activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 3199
 - ~SessionInfoMarshaller, 3201
 - createObject, 3201
 - getDataStructureType, 3201
 - looseMarshal, 3201
 - looseUnmarshal, 3201
 - SessionInfoMarshaller, 3201
 - tightMarshal1, 3202
 - tightMarshal2, 3202
- activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 3260
 - ~ShutdownInfoMarshaller, 3261
 - createObject, 3261
 - getDataStructureType, 3261
 - looseMarshal, 3261
 - looseUnmarshal, 3262
 - ShutdownInfoMarshaller, 3261
 - tightMarshal1, 3262
 - tightMarshal2, 3263
- activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 3442
 - ~SubscriptionInfoMarshaller, 3443
 - createObject, 3443
 - getDataStructureType, 3443
 - looseMarshal, 3444
 - looseUnmarshal, 3444
 - SubscriptionInfoMarshaller, 3443
 - tightMarshal1, 3444
 - tightMarshal2, 3445
- activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3576
 - ~TransactionIdMarshaller, 3577
 - looseMarshal, 3577
 - looseUnmarshal, 3578
 - tightMarshal1, 3578
 - tightMarshal2, 3579
 - tightUnmarshal, 3579
 - TransactionIdMarshaller, 3577
- activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3604
 - ~TransactionInfoMarshaller, 3604
 - createObject, 3604
 - getDataStructureType, 3604
 - looseMarshal, 3604

- looseUnmarshal, 3604
 - tightMarshal1, 3605
 - tightMarshal2, 3605
 - tightUnmarshal, 3606
 - TransactionInfoMarshaller, 3604
- activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 3743
 - ~WireFormatInfoMarshaller, 3745
 - createObject, 3745
 - getDataStructureType, 3745
 - looseMarshal, 3745
 - looseUnmarshal, 3745
 - tightMarshal1, 3746
 - tightMarshal2, 3746
 - tightUnmarshal, 3747
 - WireFormatInfoMarshaller, 3745
- activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 3781
 - ~XATransactionIdMarshaller, 3782
 - createObject, 3782
 - getDataStructureType, 3782
 - looseMarshal, 3782
 - looseUnmarshal, 3783
 - tightMarshal1, 3783
 - tightMarshal2, 3784
 - tightUnmarshal, 3784
 - XATransactionIdMarshaller, 3782
- activemq::wireformat::openwire::marshal::v2, 97
- activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 182
 - ~ActiveMQBlobMessageMarshaller, 184
 - ActiveMQBlobMessageMarshaller, 184
 - createObject, 184
 - getDataStructureType, 184
 - looseMarshal, 184
 - looseUnmarshal, 184
 - tightMarshal1, 185
 - tightMarshal2, 185
 - tightUnmarshal, 186
- activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 229
 - ~ActiveMQBytesMessageMarshaller, 231
 - ActiveMQBytesMessageMarshaller, 231
 - createObject, 231
 - getDataStructureType, 231
 - looseMarshal, 231
 - looseUnmarshal, 231
 - tightMarshal1, 232
 - tightMarshal2, 232
 - tightUnmarshal, 233
- activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 305
 - ~ActiveMQDestinationMarshaller, 307
- ActiveMQDestinationMarshaller, 307
 - looseMarshal, 307
 - looseUnmarshal, 307
 - tightMarshal1, 308
 - tightMarshal2, 308
- activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 344
 - ~ActiveMQMapMessageMarshaller, 346
 - ActiveMQMapMessageMarshaller, 346
 - createObject, 346
 - getDataStructureType, 346
 - looseMarshal, 346
 - looseUnmarshal, 346
 - tightMarshal1, 347
 - tightMarshal2, 347
- activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 371
 - ~ActiveMQMessageMarshaller, 372
 - ActiveMQMessageMarshaller, 372
 - createObject, 372
 - getDataStructureType, 372
 - looseMarshal, 372
 - looseUnmarshal, 373
 - tightMarshal1, 373
 - tightMarshal2, 374
 - tightUnmarshal, 374
- activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 416
 - ~ActiveMQObjectMessageMarshaller, 416
 - ActiveMQObjectMessageMarshaller, 416
 - createObject, 416
 - getDataStructureType, 416
 - looseMarshal, 416
 - looseUnmarshal, 417
 - tightMarshal1, 417
 - tightMarshal2, 418
 - tightUnmarshal, 418
- activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMessageMarshaller, 457
 - ~ActiveMQQueueMessageMarshaller, 458
 - ActiveMQQueueMessageMarshaller, 458
 - createObject, 458
 - getDataStructureType, 458
 - looseMarshal, 458
 - looseUnmarshal, 459
 - tightMarshal1, 459
 - tightMarshal2, 460
 - tightUnmarshal, 460
- activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 515
 - ~ActiveMQStreamMessageMarshaller, 517
 - ActiveMQStreamMessageMarshaller, 517

- createObject, 517
- getDataStructureType, 517
- looseMarshal, 517
- looseUnmarshal, 517
- tightMarshal1, 518
- tightMarshal2, 518
- tightUnmarshal, 519
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 541
 - ~ActiveMQTempDestinationMarshaller, 542
 - ActiveMQTempDestinationMarshaller, 542
 - looseMarshal, 542
 - looseUnmarshal, 543
 - tightMarshal1, 543
 - tightMarshal2, 544
 - tightUnmarshal, 544
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 568
 - ~ActiveMQTempQueueMarshaller, 570
 - ActiveMQTempQueueMarshaller, 570
 - createObject, 570
 - getDataStructureType, 570
 - looseMarshal, 570
 - looseUnmarshal, 570
 - tightMarshal1, 571
 - tightMarshal2, 571
 - tightUnmarshal, 572
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 596
 - ~ActiveMQTempTopicMarshaller, 598
 - ActiveMQTempTopicMarshaller, 598
 - createObject, 598
 - getDataStructureType, 598
 - looseMarshal, 598
 - looseUnmarshal, 598
 - tightMarshal1, 599
 - tightMarshal2, 599
 - tightUnmarshal, 600
- activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 629
 - ~ActiveMQTextMessageMarshaller, 630
 - ActiveMQTextMessageMarshaller, 630
 - createObject, 630
 - getDataStructureType, 630
 - looseMarshal, 630
 - looseUnmarshal, 631
 - tightMarshal1, 631
 - tightMarshal2, 632
 - tightUnmarshal, 632
- activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 656
 - ~ActiveMQTopicMarshaller, 658
 - ActiveMQTopicMarshaller, 658
 - createObject, 658
 - getDataStructureType, 658
 - looseMarshal, 658
 - looseUnmarshal, 658
 - tightMarshal1, 659
 - tightMarshal2, 659
 - tightUnmarshal, 660
- activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 734
 - ~BaseCommandMarshaller, 736
 - BaseCommandMarshaller, 736
 - looseMarshal, 736
 - looseUnmarshal, 737
 - tightMarshal1, 738
 - tightMarshal2, 739
 - tightUnmarshal, 740
- activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 820
 - ~BrokerIdMarshaller, 822
 - BrokerIdMarshaller, 822
 - createObject, 822
 - getDataStructureType, 822
 - looseMarshal, 822
 - looseUnmarshal, 822
 - tightMarshal1, 823
 - tightMarshal2, 823
 - tightUnmarshal, 824
- activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 853
 - ~BrokerInfoMarshaller, 853
 - BrokerInfoMarshaller, 853
 - createObject, 853
 - getDataStructureType, 853
 - looseMarshal, 853
 - looseUnmarshal, 853
 - tightMarshal1, 854
 - tightMarshal2, 854
 - tightUnmarshal, 855
- activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1196
 - ~ConnectionControlMarshaller, 1198
 - ConnectionControlMarshaller, 1198
 - createObject, 1198
 - getDataStructureType, 1198
 - looseMarshal, 1198
 - looseUnmarshal, 1198
 - tightMarshal1, 1199
 - tightMarshal2, 1199
 - tightUnmarshal, 1200
- activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1204
 - ~ConnectionErrorMarshaller, 1205
 - ConnectionErrorMarshaller, 1205
 - createObject, 1205

- getDataStructureType, 1205
 - looseMarshal, 1206
 - looseUnmarshal, 1206
 - tightMarshal1, 1206
 - tightMarshal2, 1207
 - tightUnmarshal, 1207
- activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1234
 - ~ConnectionIdMarshaller, 1235
 - ConnectionIdMarshaller, 1235
 - createObject, 1235
 - getDataStructureType, 1235
 - looseMarshal, 1235
 - looseUnmarshal, 1236
 - tightMarshal1, 1236
 - tightMarshal2, 1237
 - tightUnmarshal, 1237
- activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1263
 - ~ConnectionInfoMarshaller, 1265
 - ConnectionInfoMarshaller, 1265
 - createObject, 1265
 - getDataStructureType, 1265
 - looseMarshal, 1265
 - looseUnmarshal, 1265
 - tightMarshal1, 1266
 - tightMarshal2, 1266
 - tightUnmarshal, 1267
- activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1306
 - ~ConsumerControlMarshaller, 1308
 - ConsumerControlMarshaller, 1308
 - createObject, 1308
 - getDataStructureType, 1308
 - looseMarshal, 1308
 - looseUnmarshal, 1308
 - tightMarshal1, 1309
 - tightMarshal2, 1309
 - tightUnmarshal, 1310
- activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1334
 - ~ConsumerIdMarshaller, 1335
 - ConsumerIdMarshaller, 1335
 - createObject, 1335
 - getDataStructureType, 1335
 - looseMarshal, 1336
 - looseUnmarshal, 1336
 - tightMarshal1, 1336
 - tightMarshal2, 1337
 - tightUnmarshal, 1337
- activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1366
 - ~ConsumerInfoMarshaller, 1367
 - ConsumerInfoMarshaller, 1367
 - createObject, 1367
 - getDataStructureType, 1367
 - looseMarshal, 1367
 - looseUnmarshal, 1368
 - tightMarshal1, 1368
 - tightMarshal2, 1368
 - tightUnmarshal, 1369
- activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1393
 - ~ControlCommandMarshaller, 1394
 - ControlCommandMarshaller, 1394
 - createObject, 1394
 - getDataStructureType, 1395
 - looseMarshal, 1395
 - looseUnmarshal, 1395
 - tightMarshal1, 1395
 - tightMarshal2, 1396
 - tightUnmarshal, 1396
- activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1426
 - ~DataArrayResponseMarshaller, 1427
 - createObject, 1427
 - DataArrayResponseMarshaller, 1427
 - getDataStructureType, 1427
 - looseMarshal, 1427
 - looseUnmarshal, 1428
 - tightMarshal1, 1428
 - tightMarshal2, 1429
 - tightUnmarshal, 1429
- activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1487
 - ~DataResponseMarshaller, 1489
 - createObject, 1489
 - DataResponseMarshaller, 1489
 - getDataStructureType, 1489
 - looseMarshal, 1489
 - looseUnmarshal, 1490
 - tightMarshal1, 1490
 - tightMarshal2, 1490
 - tightUnmarshal, 1491
- activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1618
 - ~DestinationInfoMarshaller, 1619
 - createObject, 1619
 - DestinationInfoMarshaller, 1619
 - getDataStructureType, 1619
 - looseMarshal, 1620
 - looseUnmarshal, 1620
 - tightMarshal1, 1620
 - tightMarshal2, 1621
 - tightUnmarshal, 1621
- activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1651
 - ~DiscoveryEventMarshaller, 1652

- createObject, 1652
- DiscoveryEventMarshaller, 1652
- getDataStructureType, 1652
- looseMarshal, 1652
- looseUnmarshal, 1653
- tightMarshal1, 1653
- tightMarshal2, 1653
- tightUnmarshal, 1654
- activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1726
 - ~ExceptionResponseMarshaller, 1728
 - createObject, 1728
 - ExceptionResponseMarshaller, 1728
 - getDataStructureType, 1728
 - looseMarshal, 1728
 - looseUnmarshal, 1729
 - tightMarshal1, 1729
 - tightMarshal2, 1729
 - tightUnmarshal, 1730
- activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 1818
 - ~FlushCommandMarshaller, 1820
 - createObject, 1820
 - FlushCommandMarshaller, 1820
 - getDataStructureType, 1820
 - looseMarshal, 1820
 - looseUnmarshal, 1820
 - tightMarshal1, 1821
 - tightMarshal2, 1821
 - tightUnmarshal, 1822
- activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 1962
 - ~IntegerResponseMarshaller, 1964
 - createObject, 1964
 - getDataStructureType, 1964
 - IntegerResponseMarshaller, 1964
 - looseMarshal, 1964
 - looseUnmarshal, 1965
 - tightMarshal1, 1965
 - tightMarshal2, 1965
 - tightUnmarshal, 1966
- activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 2021
 - ~JournalQueueAckMarshaller, 2022
 - createObject, 2022
 - getDataStructureType, 2022
 - JournalQueueAckMarshaller, 2022
 - looseMarshal, 2022
 - looseUnmarshal, 2023
 - tightMarshal1, 2023
 - tightMarshal2, 2024
 - tightUnmarshal, 2024
- activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 2049
 - ~JournalTopicAckMarshaller, 2051
 - createObject, 2051
 - getDataStructureType, 2051
 - JournalTopicAckMarshaller, 2051
 - looseMarshal, 2051
 - looseUnmarshal, 2051
 - tightMarshal1, 2052
 - tightMarshal2, 2052
 - tightUnmarshal, 2053
- activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 2072
 - ~JournalTraceMarshaller, 2073
 - createObject, 2073
 - getDataStructureType, 2073
 - JournalTraceMarshaller, 2073
 - looseMarshal, 2073
 - looseUnmarshal, 2074
 - tightMarshal1, 2074
 - tightMarshal2, 2075
 - tightUnmarshal, 2075
- activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2102
 - ~JournalTransactionMarshaller, 2104
 - createObject, 2104
 - getDataStructureType, 2104
 - JournalTransactionMarshaller, 2104
 - looseMarshal, 2104
 - looseUnmarshal, 2104
 - tightMarshal1, 2105
 - tightMarshal2, 2105
 - tightUnmarshal, 2106
- activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 2129
 - ~KeepAliveInfoMarshaller, 2130
 - createObject, 2130
 - getDataStructureType, 2130
 - KeepAliveInfoMarshaller, 2130
 - looseMarshal, 2130
 - looseUnmarshal, 2131
 - tightMarshal1, 2131
 - tightMarshal2, 2132
 - tightUnmarshal, 2132
- activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 2166
 - ~LastPartialCommandMarshaller, 2167
 - createObject, 2167
 - getDataStructureType, 2167
 - LastPartialCommandMarshaller, 2167
 - looseMarshal, 2168
 - looseUnmarshal, 2168
 - tightMarshal1, 2168
 - tightMarshal2, 2169
 - tightUnmarshal, 2169

- activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2208
 - ~LocalTransactionIdMarshaller, 2209
 - createObject, 2209
 - getDataStructureType, 2209
 - LocalTransactionIdMarshaller, 2209
 - looseMarshal, 2209
 - looseUnmarshal, 2210
 - tightMarshal1, 2210
 - tightMarshal2, 2211
 - tightUnmarshal, 2211
- activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2334
 - ~MarshallerFactory, 2335
 - configure, 2335
- activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2403
 - ~MessageAckMarshaller, 2404
 - createObject, 2404
 - getDataStructureType, 2404
 - looseMarshal, 2404
 - looseUnmarshal, 2405
 - MessageAckMarshaller, 2404
 - tightMarshal1, 2405
 - tightMarshal2, 2406
 - tightUnmarshal, 2406
- activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2438
 - ~MessageDispatchMarshaller, 2439
 - createObject, 2439
 - getDataStructureType, 2439
 - looseMarshal, 2439
 - looseUnmarshal, 2440
 - MessageDispatchMarshaller, 2439
 - tightMarshal1, 2440
 - tightMarshal2, 2441
 - tightUnmarshal, 2441
- activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2470
 - ~MessageDispatchNotificationMarshaller, 2472
 - createObject, 2472
 - getDataStructureType, 2472
 - looseMarshal, 2472
 - looseUnmarshal, 2472
 - MessageDispatchNotificationMarshaller, 2472
 - tightMarshal1, 2473
 - tightMarshal2, 2473
 - tightUnmarshal, 2474
- activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2499
 - ~MessageIdMarshaller, 2500
 - createObject, 2500
 - getDataStructureType, 2500
 - looseMarshal, 2501
 - looseUnmarshal, 2501
 - MessageIdMarshaller, 2500
 - tightMarshal1, 2501
 - tightMarshal2, 2502
 - tightUnmarshal, 2502
- activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2532
 - ~MessageMarshaller, 2533
 - looseMarshal, 2533
 - MessageMarshaller, 2533
 - tightMarshal1, 2534
 - tightMarshal2, 2534
 - tightUnmarshal, 2535
- activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2568
 - ~MessagePullMarshaller, 2569
 - createObject, 2569
 - getDataStructureType, 2569
 - looseMarshal, 2570
 - looseUnmarshal, 2570
 - MessagePullMarshaller, 2569
 - tightMarshal1, 2570
 - tightMarshal2, 2571
 - tightUnmarshal, 2571
- activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2617
 - ~NetworkBridgeFilterMarshaller, 2618
 - createObject, 2618
 - getDataStructureType, 2618
 - looseMarshal, 2618
 - looseUnmarshal, 2619
 - NetworkBridgeFilterMarshaller, 2618
 - tightMarshal1, 2619
 - tightMarshal2, 2619
 - tightUnmarshal, 2620
- activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2735
 - ~PartialCommandMarshaller, 2736
 - createObject, 2736
 - getDataStructureType, 2737
 - looseMarshal, 2737
 - looseUnmarshal, 2737
 - PartialCommandMarshaller, 2736
 - tightMarshal1, 2738
 - tightMarshal2, 2738
 - tightUnmarshal, 2739
- activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 2842
 - ~ProducerAckMarshaller, 2844
 - createObject, 2844
 - getDataStructureType, 2844

- looseMarshal, 2844
- looseUnmarshal, 2844
- ProducerAckMarshaller, 2844
- tightMarshal1, 2845
- tightMarshal2, 2845
- tightUnmarshal, 2846
- activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 2874
 - ~ProducerIdMarshaller, 2875
 - createObject, 2875
 - getDataSetType, 2875
 - looseMarshal, 2875
 - looseUnmarshal, 2876
 - ProducerIdMarshaller, 2875
 - tightMarshal1, 2876
 - tightMarshal2, 2877
 - tightUnmarshal, 2877
- activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 2906
 - ~ProducerInfoMarshaller, 2907
 - createObject, 2907
 - getDataSetType, 2907
 - looseMarshal, 2907
 - looseUnmarshal, 2908
 - ProducerInfoMarshaller, 2907
 - tightMarshal1, 2908
 - tightMarshal2, 2909
 - tightUnmarshal, 2909
- activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 2991
 - ~RemoveInfoMarshaller, 2992
 - createObject, 2992
 - getDataSetType, 2993
 - looseMarshal, 2993
 - looseUnmarshal, 2993
 - RemoveInfoMarshaller, 2992
 - tightMarshal1, 2993
 - tightMarshal2, 2994
 - tightUnmarshal, 2994
- activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3027
 - ~RemoveSubscriptionInfoMarshaller, 3028
 - createObject, 3028
 - getDataSetType, 3028
 - looseMarshal, 3028
 - looseUnmarshal, 3029
 - RemoveSubscriptionInfoMarshaller, 3028
 - tightMarshal1, 3029
 - tightMarshal2, 3030
 - tightUnmarshal, 3030
- activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 3054
 - ~ReplayCommandMarshaller, 3055
 - createObject, 3055
 - getDataSetType, 3055
 - looseMarshal, 3055
 - looseUnmarshal, 3056
 - ReplayCommandMarshaller, 3055
 - tightMarshal1, 3056
 - tightMarshal2, 3057
 - tightUnmarshal, 3057
- activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 3089
 - ~ResponseMarshaller, 3090
 - createObject, 3090
 - getDataSetType, 3090
 - looseMarshal, 3091
 - looseUnmarshal, 3091
 - ResponseMarshaller, 3090
 - tightMarshal1, 3092
 - tightMarshal2, 3092
- activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 3165
 - ~SessionIdMarshaller, 3166
 - createObject, 3166
 - getDataSetType, 3166
 - looseMarshal, 3167
 - looseUnmarshal, 3167
 - SessionIdMarshaller, 3166
 - tightMarshal1, 3167
 - tightMarshal2, 3168
- activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 3207
 - ~SessionInfoMarshaller, 3209
 - createObject, 3209
 - getDataSetType, 3209
 - looseMarshal, 3209
 - looseUnmarshal, 3209
 - SessionInfoMarshaller, 3209
 - tightMarshal1, 3210
 - tightMarshal2, 3210
- activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 3256
 - ~ShutdownInfoMarshaller, 3257
 - createObject, 3257
 - getDataSetType, 3257
 - looseMarshal, 3257
 - looseUnmarshal, 3258
 - ShutdownInfoMarshaller, 3257
 - tightMarshal1, 3258
 - tightMarshal2, 3259
- activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 3457
 - ~SubscriptionInfoMarshaller, 3459

- createObject, 3459
- getDataStructureType, 3459
- looseMarshal, 3459
- looseUnmarshal, 3459
- SubscriptionInfoMarshaller, 3459
- tightMarshal1, 3460
- tightMarshal2, 3460
- tightUnmarshal, 3461
- activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3580
 - ~TransactionIdMarshaller, 3581
 - looseMarshal, 3581
 - looseUnmarshal, 3581
 - tightMarshal1, 3582
 - tightMarshal2, 3582
 - tightUnmarshal, 3583
 - TransactionIdMarshaller, 3581
- activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3618
 - ~TransactionInfoMarshaller, 3620
 - createObject, 3620
 - getDataStructureType, 3620
 - looseMarshal, 3620
 - looseUnmarshal, 3620
 - tightMarshal1, 3621
 - tightMarshal2, 3621
 - tightUnmarshal, 3622
 - TransactionInfoMarshaller, 3620
- activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 3735
 - ~WireFormatInfoMarshaller, 3737
 - createObject, 3737
 - getDataStructureType, 3737
 - looseMarshal, 3737
 - looseUnmarshal, 3737
 - tightMarshal1, 3738
 - tightMarshal2, 3738
 - tightUnmarshal, 3739
 - WireFormatInfoMarshaller, 3737
- activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 3773
 - ~XATransactionIdMarshaller, 3774
 - createObject, 3774
 - getDataStructureType, 3774
 - looseMarshal, 3774
 - looseUnmarshal, 3775
 - tightMarshal1, 3775
 - tightMarshal2, 3776
 - tightUnmarshal, 3776
 - XATransactionIdMarshaller, 3774
- activemq::wireformat::openwire::marshal::v3, 101
- activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller, 171
 - ~ActiveMQBlobMessageMarshaller, 172
 - ActiveMQBlobMessageMarshaller, 172
 - createObject, 172
 - getDataStructureType, 172
 - looseMarshal, 172
 - looseUnmarshal, 173
 - tightMarshal1, 173
 - tightMarshal2, 173
 - ~ActiveMQBlobMessageMarshaller, 174
- activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 210
 - ~ActiveMQBytesMessageMarshaller, 211
 - ActiveMQBytesMessageMarshaller, 211
 - createObject, 211
 - getDataStructureType, 211
 - looseMarshal, 211
 - looseUnmarshal, 212
 - ~ActiveMQBytesMessageMarshaller, 212
 - tightMarshal1, 212
 - tightMarshal2, 212
 - tightUnmarshal, 213
- activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 289
 - ~ActiveMQDestinationMarshaller, 291
 - ActiveMQDestinationMarshaller, 291
 - looseMarshal, 291
 - looseUnmarshal, 291
 - tightMarshal1, 292
 - tightMarshal2, 292
 - ~ActiveMQDestinationMarshaller, 293
- activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 328
 - ~ActiveMQMapMessageMarshaller, 330
 - ActiveMQMapMessageMarshaller, 330
 - createObject, 330
 - getDataStructureType, 330
 - looseMarshal, 330
 - looseUnmarshal, 330
 - tightMarshal1, 331
 - tightMarshal2, 331
 - ~ActiveMQMapMessageMarshaller, 332
- activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 355
 - ~ActiveMQMessageMarshaller, 356
 - ActiveMQMessageMarshaller, 356
 - createObject, 356
 - getDataStructureType, 356
 - looseMarshal, 357
 - looseUnmarshal, 357
 - tightMarshal1, 357
 - tightMarshal2, 358
 - tightUnmarshal, 358
- activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 399
 - ~ActiveMQObjectMessageMarshaller, 400

- ActiveMQObjectMessageMarshaller, 400
- createObject, 400
- getDataStructureType, 400
- looseMarshal, 400
- looseUnmarshal, 401
- tightMarshal1, 401
- tightMarshal2, 402
- tightUnmarshal, 402
- activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMessageMarshaller, 441
 - ~ActiveMQQueueMarshaller, 442
 - ActiveMQQueueMarshaller, 442
 - createObject, 442
 - getDataStructureType, 442
 - looseMarshal, 442
 - looseUnmarshal, 443
 - tightMarshal1, 443
 - tightMarshal2, 444
 - tightUnmarshal, 444
- activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 499
 - ~ActiveMQStreamMessageMarshaller, 501
 - ActiveMQStreamMessageMarshaller, 501
 - createObject, 501
 - getDataStructureType, 501
 - looseMarshal, 501
 - looseUnmarshal, 501
 - tightMarshal1, 502
 - tightMarshal2, 502
 - tightUnmarshal, 503
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 527
 - ~ActiveMQTempDestinationMarshaller, 528
 - ActiveMQTempDestinationMarshaller, 528
 - looseMarshal, 528
 - looseUnmarshal, 528
 - tightMarshal1, 529
 - tightMarshal2, 529
 - tightUnmarshal, 530
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 552
 - ~ActiveMQTempQueueMarshaller, 554
 - ActiveMQTempQueueMarshaller, 554
 - createObject, 554
 - getDataStructureType, 554
 - looseMarshal, 554
 - looseUnmarshal, 554
 - tightMarshal1, 555
 - tightMarshal2, 555
 - tightUnmarshal, 556
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 580
 - ~ActiveMQTempTopicMarshaller, 582
- ActiveMQTempTopicMarshaller, 582
- createObject, 582
- getDataStructureType, 582
- looseMarshal, 582
- looseUnmarshal, 582
- tightMarshal1, 583
- tightMarshal2, 583
- tightUnmarshal, 584
- activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 609
 - ~ActiveMQTextMessageMarshaller, 610
 - ActiveMQTextMessageMarshaller, 610
 - createObject, 610
 - getDataStructureType, 610
 - looseMarshal, 611
 - looseUnmarshal, 611
 - tightMarshal1, 611
 - tightMarshal2, 612
 - tightUnmarshal, 612
- activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 636
 - ~ActiveMQTopicMarshaller, 638
 - ActiveMQTopicMarshaller, 638
 - createObject, 638
 - getDataStructureType, 638
 - looseMarshal, 638
 - looseUnmarshal, 638
 - tightMarshal1, 639
 - tightMarshal2, 639
 - tightUnmarshal, 640
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempDefinitionMarshaller, 701
 - ~BaseCommandMarshaller, 702
 - BaseCommandMarshaller, 702
 - looseMarshal, 702
 - looseUnmarshal, 703
 - tightMarshal1, 704
 - tightMarshal2, 705
 - tightUnmarshal, 707
- activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 800
 - ~BrokerIdMarshaller, 802
 - BrokerIdMarshaller, 802
 - createObject, 802
 - getDataStructureType, 802
 - looseMarshal, 802
 - looseUnmarshal, 803
 - tightMarshal1, 803
 - tightMarshal2, 804
 - tightUnmarshal, 804
- activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 831
 - ~BrokerInfoMarshaller, 833
 - BrokerInfoMarshaller, 833

- createObject, 833
- getDataStructureType, 833
- looseMarshal, 833
- looseUnmarshal, 833
- tightMarshal1, 834
- tightMarshal2, 834
- tightUnmarshal, 835
- activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1177
 - ~ConnectionControlMarshaller, 1178
 - ConnectionControlMarshaller, 1178
 - createObject, 1178
 - getDataStructureType, 1178
 - looseMarshal, 1178
 - looseUnmarshal, 1179
 - tightMarshal1, 1179
 - tightMarshal2, 1179
 - tightUnmarshal, 1180
- activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1208
 - ~ConnectionErrorMarshaller, 1209
 - ConnectionErrorMarshaller, 1209
 - createObject, 1209
 - getDataStructureType, 1209
 - looseMarshal, 1210
 - looseUnmarshal, 1210
 - tightMarshal1, 1210
 - tightMarshal2, 1211
 - tightUnmarshal, 1211
- activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1238
 - ~ConnectionIdMarshaller, 1239
 - ConnectionIdMarshaller, 1239
 - createObject, 1239
 - getDataStructureType, 1239
 - looseMarshal, 1239
 - looseUnmarshal, 1240
 - tightMarshal1, 1240
 - tightMarshal2, 1240
 - tightUnmarshal, 1241
- activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1267
 - ~ConnectionInfoMarshaller, 1269
 - ConnectionInfoMarshaller, 1269
 - createObject, 1269
 - getDataStructureType, 1269
 - looseMarshal, 1269
 - looseUnmarshal, 1269
 - tightMarshal1, 1270
 - tightMarshal2, 1270
 - tightUnmarshal, 1271
- activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1310
 - ~ConsumerControlMarshaller, 1312
 - ConsumerControlMarshaller, 1312
 - createObject, 1312
 - getDataStructureType, 1312
 - looseMarshal, 1312
 - looseUnmarshal, 1312
 - tightMarshal1, 1313
 - tightMarshal2, 1313
- activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1338
 - ~ConsumerIdMarshaller, 1339
 - ConsumerIdMarshaller, 1339
 - createObject, 1339
 - getDataStructureType, 1339
 - looseMarshal, 1339
 - looseUnmarshal, 1340
 - tightMarshal1, 1340
 - tightMarshal2, 1341
- activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1369
 - ~ConsumerInfoMarshaller, 1371
 - ConsumerInfoMarshaller, 1371
 - createObject, 1371
 - getDataStructureType, 1371
 - looseMarshal, 1371
 - looseUnmarshal, 1371
 - tightMarshal1, 1372
 - tightMarshal2, 1372
- activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1397
 - ~ControlCommandMarshaller, 1398
 - ControlCommandMarshaller, 1398
 - createObject, 1398
 - getDataStructureType, 1398
 - looseMarshal, 1399
 - looseUnmarshal, 1399
 - tightMarshal1, 1399
 - tightMarshal2, 1400
- activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1430
 - ~DataArrayResponseMarshaller, 1431
 - createObject, 1431
 - DataArrayResponseMarshaller, 1431
 - getDataStructureType, 1431
 - looseMarshal, 1431
 - looseUnmarshal, 1432
 - tightMarshal1, 1432
 - tightMarshal2, 1433
- activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1491
 - ~DataResponseMarshaller, 1492
 - DataResponseMarshaller, 1492
 - createObject, 1492
 - getDataStructureType, 1492
 - looseMarshal, 1492
 - looseUnmarshal, 1493
 - tightMarshal1, 1493
 - tightMarshal2, 1494

- ~DataResponseMarshaller, 1493
- createObject, 1493
- DataResponseMarshaller, 1493
- getDataStructureType, 1493
- looseMarshal, 1493
- looseUnmarshal, 1494
- tightMarshal1, 1494
- tightMarshal2, 1494
- tightUnmarshal, 1495
- activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1622
 - ~DestinationInfoMarshaller, 1623
 - createObject, 1623
 - DestinationInfoMarshaller, 1623
 - getDataStructureType, 1623
 - looseMarshal, 1623
 - looseUnmarshal, 1624
 - tightMarshal1, 1624
 - tightMarshal2, 1625
 - tightUnmarshal, 1625
- activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1654
 - ~DiscoveryEventMarshaller, 1656
 - createObject, 1656
 - DiscoveryEventMarshaller, 1656
 - getDataStructureType, 1656
 - looseMarshal, 1656
 - looseUnmarshal, 1656
 - tightMarshal1, 1657
 - tightMarshal2, 1657
 - tightUnmarshal, 1658
- activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1730
 - ~ExceptionResponseMarshaller, 1732
 - createObject, 1732
 - ExceptionResponseMarshaller, 1732
 - getDataStructureType, 1732
 - looseMarshal, 1732
 - looseUnmarshal, 1733
 - tightMarshal1, 1733
 - tightMarshal2, 1733
 - tightUnmarshal, 1734
- activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 1822
 - ~FlushCommandMarshaller, 1824
 - createObject, 1824
 - FlushCommandMarshaller, 1824
 - getDataStructureType, 1824
 - looseMarshal, 1824
 - looseUnmarshal, 1824
 - tightMarshal1, 1825
 - tightMarshal2, 1825
 - tightUnmarshal, 1826
- activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 1966
 - ~IntegerResponseMarshaller, 1968
 - createObject, 1968
 - getDataStructureType, 1968
 - IntegerResponseMarshaller, 1968
 - looseMarshal, 1968
 - looseUnmarshal, 1969
 - tightMarshal1, 1969
 - tightMarshal2, 1969
 - tightUnmarshal, 1970
- activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 2028
 - ~JournalQueueAckMarshaller, 2030
 - createObject, 2030
 - getDataStructureType, 2030
 - JournalQueueAckMarshaller, 2030
 - looseMarshal, 2030
 - looseUnmarshal, 2030
 - tightMarshal1, 2031
 - tightMarshal2, 2031
 - tightUnmarshal, 2032
- activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 2053
 - ~JournalTopicAckMarshaller, 2055
 - createObject, 2055
 - getDataStructureType, 2055
 - JournalTopicAckMarshaller, 2055
 - looseMarshal, 2055
 - looseUnmarshal, 2055
 - tightMarshal1, 2056
 - tightMarshal2, 2056
 - tightUnmarshal, 2057
- activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 2076
 - ~JournalTraceMarshaller, 2077
 - createObject, 2077
 - getDataStructureType, 2077
 - JournalTraceMarshaller, 2077
 - looseMarshal, 2077
 - looseUnmarshal, 2078
 - tightMarshal1, 2078
 - tightMarshal2, 2078
 - tightUnmarshal, 2079
- activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 2106
 - ~JournalTransactionMarshaller, 2108
 - createObject, 2108
 - getDataStructureType, 2108
 - JournalTransactionMarshaller, 2108
 - looseMarshal, 2108
 - looseUnmarshal, 2108
 - tightMarshal1, 2109
 - tightMarshal2, 2109

- tightUnmarshal, 2110
- activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 2133
 - ~KeepAliveInfoMarshaller, 2134
 - createObject, 2134
 - getDataStructureType, 2134
 - KeepAliveInfoMarshaller, 2134
 - looseMarshal, 2134
 - looseUnmarshal, 2135
 - tightMarshal1, 2135
 - tightMarshal2, 2136
 - tightUnmarshal, 2136
- activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 2162
 - ~LastPartialCommandMarshaller, 2163
 - createObject, 2163
 - getDataStructureType, 2163
 - LastPartialCommandMarshaller, 2163
 - looseMarshal, 2164
 - looseUnmarshal, 2164
 - tightMarshal1, 2164
 - tightMarshal2, 2165
 - tightUnmarshal, 2165
- activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2212
 - ~LocalTransactionIdMarshaller, 2213
 - createObject, 2213
 - getDataStructureType, 2213
 - LocalTransactionIdMarshaller, 2213
 - looseMarshal, 2213
 - looseUnmarshal, 2214
 - tightMarshal1, 2214
 - tightMarshal2, 2215
 - tightUnmarshal, 2215
- activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 2331
 - ~MarshallerFactory, 2332
 - configure, 2332
- activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2407
 - ~MessageAckMarshaller, 2408
 - createObject, 2408
 - getDataStructureType, 2408
 - looseMarshal, 2408
 - looseUnmarshal, 2409
 - MessageAckMarshaller, 2408
 - tightMarshal1, 2409
 - tightMarshal2, 2410
 - tightUnmarshal, 2410
- activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2442
 - ~MessageDispatchMarshaller, 2443
 - createObject, 2443
 - getDataStructureType, 2443
- looseMarshal, 2443
- looseUnmarshal, 2444
- MessageDispatchMarshaller, 2443
- tightMarshal1, 2444
- tightMarshal2, 2445
- tightUnmarshal, 2445
- activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2474
 - ~MessageDispatchNotificationMarshaller, 2476
 - createObject, 2476
 - getDataStructureType, 2476
 - looseUnmarshal, 2476
 - MessageDispatchNotificationMarshaller, 2476
 - tightMarshal1, 2477
 - tightMarshal2, 2477
 - tightUnmarshal, 2478
- activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2510
 - ~MessageIdMarshaller, 2512
 - createObject, 2512
 - getDataStructureType, 2512
 - looseMarshal, 2512
 - looseUnmarshal, 2512
 - MessageIdMarshaller, 2512
 - tightMarshal1, 2513
 - tightMarshal2, 2513
 - tightUnmarshal, 2514
- activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2527
 - ~MessageMarshaller, 2529
 - looseMarshal, 2529
 - looseUnmarshal, 2529
 - MessageMarshaller, 2529
 - tightMarshal1, 2530
 - tightMarshal2, 2530
 - tightUnmarshal, 2531
- activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2576
 - ~MessagePullMarshaller, 2577
 - createObject, 2577
 - getDataStructureType, 2577
 - looseMarshal, 2577
 - looseUnmarshal, 2578
 - MessagePullMarshaller, 2577
 - tightMarshal1, 2578
 - tightMarshal2, 2579
 - tightUnmarshal, 2579
- activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2628
 - ~NetworkBridgeFilterMarshaller, 2630
 - createObject, 2630

- getDataStructureType, 2630
 - looseMarshal, 2630
 - looseUnmarshal, 2630
 - NetworkBridgeFilterMarshaller, 2630
 - tightMarshal1, 2631
 - tightMarshal2, 2631
 - tightUnmarshal, 2632
- activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2744
 - ~PartialCommandMarshaller, 2745
 - createObject, 2745
 - getDataStructureType, 2745
 - looseMarshal, 2745
 - looseUnmarshal, 2746
 - PartialCommandMarshaller, 2745
 - tightMarshal1, 2746
 - tightMarshal2, 2747
 - tightUnmarshal, 2747
- activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 2850
 - ~ProducerAckMarshaller, 2852
 - createObject, 2852
 - getDataStructureType, 2852
 - looseMarshal, 2852
 - looseUnmarshal, 2852
 - ProducerAckMarshaller, 2852
 - tightMarshal1, 2853
 - tightMarshal2, 2853
 - tightUnmarshal, 2854
- activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 2881
 - ~ProducerIdMarshaller, 2883
 - createObject, 2883
 - getDataStructureType, 2883
 - looseMarshal, 2883
 - looseUnmarshal, 2883
 - ProducerIdMarshaller, 2883
 - tightMarshal1, 2884
 - tightMarshal2, 2884
 - tightUnmarshal, 2885
- activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 2918
 - ~ProducerInfoMarshaller, 2919
 - createObject, 2919
 - getDataStructureType, 2919
 - looseMarshal, 2919
 - looseUnmarshal, 2920
 - ProducerInfoMarshaller, 2919
 - tightMarshal1, 2920
 - tightMarshal2, 2921
 - tightUnmarshal, 2921
- activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 2999
 - ~RemoveInfoMarshaller, 3000
- createObject, 3000
- getDataStructureType, 3000
- looseMarshal, 3000
- looseUnmarshal, 3001
- RemoveInfoMarshaller, 3000
- tightMarshal1, 3001
- tightMarshal2, 3002
- tightUnmarshal, 3002
- activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3023
 - ~RemoveSubscriptionInfoMarshaller, 3024
 - createObject, 3024
 - getDataStructureType, 3024
 - looseMarshal, 3025
 - looseUnmarshal, 3025
 - RemoveSubscriptionInfoMarshaller, 3024
 - tightMarshal1, 3025
 - tightMarshal2, 3026
- activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 3058
 - ~ReplayCommandMarshaller, 3059
 - createObject, 3059
 - getDataStructureType, 3059
 - looseMarshal, 3059
 - looseUnmarshal, 3060
 - ReplayCommandMarshaller, 3059
 - tightMarshal1, 3060
 - tightMarshal2, 3061
- activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 3098
 - ~ResponseMarshaller, 3099
 - createObject, 3099
 - getDataStructureType, 3099
 - looseMarshal, 3100
 - looseUnmarshal, 3100
 - ResponseMarshaller, 3099
 - tightMarshal1, 3101
 - tightMarshal2, 3101
- activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 3180
 - ~SessionIdMarshaller, 3182
 - createObject, 3182
 - getDataStructureType, 3182
 - looseMarshal, 3182
 - looseUnmarshal, 3182
 - SessionIdMarshaller, 3182
 - tightMarshal1, 3183
 - tightMarshal2, 3183
- activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 3203
 - ~SessionInfoMarshaller, 3204

- ~SessionInfoMarshaller, 3205
- createObject, 3205
- getDataStructureType, 3205
- looseMarshal, 3205
- looseUnmarshal, 3205
- SessionInfoMarshaller, 3205
- tightMarshal1, 3206
- tightMarshal2, 3206
- tightUnmarshal, 3207
- activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 3268
 - ~ShutdownInfoMarshaller, 3269
 - createObject, 3269
 - getDataStructureType, 3269
 - looseMarshal, 3269
 - looseUnmarshal, 3270
 - ShutdownInfoMarshaller, 3269
 - tightMarshal1, 3270
 - tightMarshal2, 3271
 - tightUnmarshal, 3271
- activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 3438
 - ~SubscriptionInfoMarshaller, 3439
 - createObject, 3439
 - getDataStructureType, 3440
 - looseMarshal, 3440
 - looseUnmarshal, 3440
 - SubscriptionInfoMarshaller, 3439
 - tightMarshal1, 3441
 - tightMarshal2, 3441
 - tightUnmarshal, 3441
- activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3583
 - ~TransactionIdMarshaller, 3585
 - looseMarshal, 3585
 - looseUnmarshal, 3585
 - tightMarshal1, 3586
 - tightMarshal2, 3586
 - tightUnmarshal, 3587
 - TransactionIdMarshaller, 3585
- activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3606
 - ~TransactionInfoMarshaller, 3608
 - createObject, 3608
 - getDataStructureType, 3608
 - looseMarshal, 3608
 - looseUnmarshal, 3608
 - tightMarshal1, 3609
 - tightMarshal2, 3609
 - tightUnmarshal, 3610
 - TransactionInfoMarshaller, 3608
- activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 3747
 - ~WireFormatInfoMarshaller, 3749
 - createObject, 3749
 - getDataStructureType, 3749
 - looseMarshal, 3749
 - looseUnmarshal, 3749
 - tightMarshal1, 3750
 - tightMarshal2, 3750
 - tightUnmarshal, 3751
 - WireFormatInfoMarshaller, 3749
- activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 3785
 - ~XATransactionIdMarshaller, 3786
 - createObject, 3786
 - getDataStructureType, 3786
 - looseMarshal, 3786
 - looseUnmarshal, 3787
 - tightMarshal1, 3787
 - tightMarshal2, 3788
 - tightUnmarshal, 3788
 - XATransactionIdMarshaller, 3786
- activemq::wireformat::openwire::marshal::v4, 178
 - ~ActiveMQBlobMessageMarshaller, 180
 - ActiveMQBlobMessageMarshaller, 180
 - createObject, 180
 - getDataStructureType, 180
 - looseMarshal, 180
 - looseUnmarshal, 180
 - tightMarshal1, 181
 - tightMarshal2, 181
 - tightUnmarshal, 182
- activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 178
 - ~ActiveMQBlobMessageMarshaller, 180
 - ActiveMQBlobMessageMarshaller, 180
 - createObject, 180
 - getDataStructureType, 180
 - looseMarshal, 180
 - looseUnmarshal, 180
 - tightMarshal1, 181
 - tightMarshal2, 181
 - tightUnmarshal, 182
- activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 217
 - ~ActiveMQBytesMessageMarshaller, 219
 - ActiveMQBytesMessageMarshaller, 219
 - createObject, 219
 - getDataStructureType, 219
 - looseMarshal, 219
 - looseUnmarshal, 219
 - tightMarshal1, 220
 - tightMarshal2, 220
 - tightUnmarshal, 221
- activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 297
 - ~ActiveMQDestinationMarshaller, 299
 - ActiveMQDestinationMarshaller, 299
 - looseMarshal, 299
 - looseUnmarshal, 299
 - tightMarshal1, 300
 - tightMarshal2, 300
 - tightUnmarshal, 301
- activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 336
 - ~ActiveMQMapMessageMarshaller, 337
 - ActiveMQMapMessageMarshaller, 337
 - createObject, 337
 - getDataStructureType, 337
 - looseMarshal, 337
 - looseUnmarshal, 337
 - tightMarshal1, 338
 - tightMarshal2, 338
 - tightUnmarshal, 338

- ~ActiveMQMapMessageMarshaller, 338
- ActiveMQMapMessageMarshaller, 338
- createObject, 338
- getDataStructureType, 338
- looseMarshal, 338
- looseUnmarshal, 338
- tightMarshal1, 339
- tightMarshal2, 339
- tightUnmarshal, 340
- activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 363
 - ~ActiveMQMessageMarshaller, 364
 - ActiveMQMessageMarshaller, 364
 - createObject, 364
 - getDataStructureType, 364
 - looseMarshal, 364
 - looseUnmarshal, 365
 - tightMarshal1, 365
 - tightMarshal2, 366
 - tightUnmarshal, 366
- activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 407
 - ~ActiveMQObjectMessageMarshaller, 408
 - ActiveMQObjectMessageMarshaller, 408
 - createObject, 408
 - getDataStructureType, 408
 - looseMarshal, 408
 - looseUnmarshal, 409
 - tightMarshal1, 409
 - tightMarshal2, 410
 - tightUnmarshal, 410
- activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller, 449
 - ~ActiveMQQueueMarshaller, 450
 - ActiveMQQueueMarshaller, 450
 - createObject, 450
 - getDataStructureType, 450
 - looseMarshal, 450
 - looseUnmarshal, 451
 - tightMarshal1, 451
 - tightMarshal2, 452
 - tightUnmarshal, 452
- activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 507
 - ~ActiveMQStreamMessageMarshaller, 509
 - ActiveMQStreamMessageMarshaller, 509
 - createObject, 509
 - getDataStructureType, 509
 - looseMarshal, 509
 - looseUnmarshal, 509
 - tightMarshal1, 510
 - tightMarshal2, 510
 - tightUnmarshal, 511
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 534
 - ~ActiveMQTempDestinationMarshaller, 535
 - ActiveMQTempDestinationMarshaller, 535
 - looseMarshal, 535
 - looseUnmarshal, 535
 - tightMarshal1, 536
 - tightMarshal2, 536
 - tightUnmarshal, 537
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 560
 - ~ActiveMQTempQueueMarshaller, 562
 - ActiveMQTempQueueMarshaller, 562
 - createObject, 562
 - getDataStructureType, 562
 - looseMarshal, 562
 - looseUnmarshal, 562
 - tightMarshal1, 563
 - tightMarshal2, 563
 - tightUnmarshal, 564
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 584
 - ~ActiveMQTempTopicMarshaller, 586
 - ActiveMQTempTopicMarshaller, 586
 - createObject, 586
 - getDataStructureType, 586
 - looseMarshal, 586
 - looseUnmarshal, 586
 - tightMarshal1, 587
 - tightMarshal2, 587
 - tightUnmarshal, 588
- activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 613
 - ~ActiveMQTextMessageMarshaller, 614
 - ActiveMQTextMessageMarshaller, 614
 - createObject, 614
 - getDataStructureType, 614
 - looseMarshal, 615
 - looseUnmarshal, 615
 - tightMarshal1, 615
 - tightMarshal2, 616
 - tightUnmarshal, 616
- activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 640
 - ~ActiveMQTopicMarshaller, 642
 - ActiveMQTopicMarshaller, 642
 - createObject, 642
 - getDataStructureType, 642
 - looseMarshal, 642
 - looseUnmarshal, 642
 - tightMarshal1, 643
 - tightMarshal2, 643
 - tightUnmarshal, 644

- activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 708
 - ~BaseCommandMarshaller, 709
 - BaseCommandMarshaller, 709
 - looseMarshal, 709
 - looseUnmarshal, 710
 - tightMarshal1, 711
 - tightMarshal2, 712
 - tightUnmarshal, 713
- activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 805
 - ~BrokerIdMarshaller, 806
 - BrokerIdMarshaller, 806
 - createObject, 806
 - getDataStructureType, 806
 - looseMarshal, 806
 - looseUnmarshal, 807
 - tightMarshal1, 807
 - tightMarshal2, 807
 - tightUnmarshal, 808
- activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 835
 - ~BrokerInfoMarshaller, 837
 - BrokerInfoMarshaller, 837
 - createObject, 837
 - getDataStructureType, 837
 - looseMarshal, 837
 - looseUnmarshal, 837
 - tightMarshal1, 838
 - tightMarshal2, 838
 - tightUnmarshal, 839
- activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1180
 - ~ConnectionControlMarshaller, 1182
 - ConnectionControlMarshaller, 1182
 - createObject, 1182
 - getDataStructureType, 1182
 - looseMarshal, 1182
 - looseUnmarshal, 1182
 - tightMarshal1, 1183
 - tightMarshal2, 1183
 - tightUnmarshal, 1184
- activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1212
 - ~ConnectionErrorMarshaller, 1213
 - ConnectionErrorMarshaller, 1213
 - createObject, 1213
 - getDataStructureType, 1213
 - looseMarshal, 1213
 - looseUnmarshal, 1214
 - tightMarshal1, 1214
 - tightMarshal2, 1215
 - tightUnmarshal, 1215
- activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1241
 - ~ConnectionIdMarshaller, 1243
 - ConnectionIdMarshaller, 1243
 - createObject, 1243
 - getDataStructureType, 1243
 - looseMarshal, 1243
 - looseUnmarshal, 1243
 - tightMarshal1, 1244
 - tightMarshal2, 1244
 - tightUnmarshal, 1245
- activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1271
 - ~ConnectionInfoMarshaller, 1273
 - ConnectionInfoMarshaller, 1273
 - createObject, 1273
 - getDataStructureType, 1273
 - looseMarshal, 1273
 - looseUnmarshal, 1273
 - tightMarshal1, 1274
 - tightMarshal2, 1274
 - tightUnmarshal, 1275
- activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1314
 - ~ConsumerControlMarshaller, 1316
 - ConsumerControlMarshaller, 1316
 - createObject, 1316
 - getDataStructureType, 1316
 - looseMarshal, 1316
 - looseUnmarshal, 1316
 - tightMarshal1, 1317
 - tightMarshal2, 1317
 - tightUnmarshal, 1318
- activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1342
 - ~ConsumerIdMarshaller, 1343
 - ConsumerIdMarshaller, 1343
 - createObject, 1343
 - getDataStructureType, 1343
 - looseMarshal, 1343
 - looseUnmarshal, 1344
 - tightMarshal1, 1344
 - tightMarshal2, 1344
 - tightUnmarshal, 1345
- activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1373
 - ~ConsumerInfoMarshaller, 1375
 - ConsumerInfoMarshaller, 1375
 - createObject, 1375
 - getDataStructureType, 1375
 - looseMarshal, 1375
 - looseUnmarshal, 1375
 - tightMarshal1, 1376
 - tightMarshal2, 1376

- tightUnmarshal, 1377
- activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1401
 - ~ControlCommandMarshaller, 1402
 - ControlCommandMarshaller, 1402
 - createObject, 1402
 - getDataStructureType, 1402
 - looseMarshal, 1402
 - looseUnmarshal, 1403
 - tightMarshal1, 1403
 - tightMarshal2, 1404
 - tightUnmarshal, 1404
- activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1434
 - ~DataArrayResponseMarshaller, 1435
 - createObject, 1435
 - DataArrayResponseMarshaller, 1435
 - getDataStructureType, 1435
 - looseMarshal, 1435
 - looseUnmarshal, 1436
 - tightMarshal1, 1436
 - tightMarshal2, 1437
 - tightUnmarshal, 1437
- activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1495
 - ~DataResponseMarshaller, 1497
 - createObject, 1497
 - DataResponseMarshaller, 1497
 - getDataStructureType, 1497
 - looseMarshal, 1497
 - looseUnmarshal, 1498
 - tightMarshal1, 1498
 - tightMarshal2, 1498
 - tightUnmarshal, 1499
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1626
 - ~DestinationInfoMarshaller, 1627
 - createObject, 1627
 - DestinationInfoMarshaller, 1627
 - getDataStructureType, 1627
 - looseMarshal, 1627
 - looseUnmarshal, 1628
 - tightMarshal1, 1628
 - tightMarshal2, 1629
 - tightUnmarshal, 1629
- activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1658
 - ~DiscoveryEventMarshaller, 1660
 - createObject, 1660
 - DiscoveryEventMarshaller, 1660
 - getDataStructureType, 1660
 - looseMarshal, 1660
 - looseUnmarshal, 1660
 - tightMarshal1, 1661
- tightMarshal2, 1661
- activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1738
 - ~ExceptionResponseMarshaller, 1740
 - createObject, 1740
 - ExceptionResponseMarshaller, 1740
 - getDataStructureType, 1740
 - looseMarshal, 1740
 - looseUnmarshal, 1741
 - tightMarshal1, 1741
 - tightMarshal2, 1741
- activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 1826
 - ~FlushCommandMarshaller, 1828
 - createObject, 1828
 - FlushCommandMarshaller, 1828
 - getDataStructureType, 1828
 - looseMarshal, 1828
 - looseUnmarshal, 1828
 - tightMarshal1, 1829
 - tightMarshal2, 1829
- activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 1970
 - ~IntegerResponseMarshaller, 1972
 - createObject, 1972
 - getDataStructureType, 1972
 - IntegerResponseMarshaller, 1972
 - looseMarshal, 1972
 - looseUnmarshal, 1973
 - tightMarshal1, 1973
 - tightMarshal2, 1973
- activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 2032
 - ~JournalQueueAckMarshaller, 2034
 - createObject, 2034
 - getDataStructureType, 2034
 - JournalQueueAckMarshaller, 2034
 - looseMarshal, 2034
 - looseUnmarshal, 2034
 - tightMarshal1, 2035
 - tightMarshal2, 2035
- activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 2061
 - ~JournalTopicAckMarshaller, 2063
 - createObject, 2063
 - getDataStructureType, 2063
 - JournalTopicAckMarshaller, 2063
 - looseMarshal, 2063
 - looseUnmarshal, 2063

- tightMarshal1, 2064
- tightMarshal2, 2064
- tightUnmarshal, 2065
- activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 2083
 - ~JournalTraceMarshaller, 2085
 - createObject, 2085
 - getDataStructureType, 2085
 - JournalTraceMarshaller, 2085
 - looseMarshal, 2085
 - looseUnmarshal, 2085
 - tightMarshal1, 2086
 - tightMarshal2, 2086
 - tightUnmarshal, 2087
- activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 2114
 - ~JournalTransactionMarshaller, 2116
 - createObject, 2116
 - getDataStructureType, 2116
 - JournalTransactionMarshaller, 2116
 - looseMarshal, 2116
 - looseUnmarshal, 2116
 - tightMarshal1, 2117
 - tightMarshal2, 2117
 - tightUnmarshal, 2118
- activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 2137
 - ~KeepAliveInfoMarshaller, 2138
 - createObject, 2138
 - getDataStructureType, 2138
 - KeepAliveInfoMarshaller, 2138
 - looseMarshal, 2138
 - looseUnmarshal, 2139
 - tightMarshal1, 2139
 - tightMarshal2, 2140
 - tightUnmarshal, 2140
- activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 2174
 - ~LastPartialCommandMarshaller, 2175
 - createObject, 2175
 - getDataStructureType, 2175
 - LastPartialCommandMarshaller, 2175
 - looseMarshal, 2176
 - looseUnmarshal, 2176
 - tightMarshal1, 2176
 - tightMarshal2, 2177
 - tightUnmarshal, 2177
- activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 2220
 - ~LocalTransactionIdMarshaller, 2221
 - createObject, 2221
 - getDataStructureType, 2221
 - LocalTransactionIdMarshaller, 2221
 - looseMarshal, 2221
 - looseUnmarshal, 2222
 - tightMarshal1, 2222
 - tightMarshal2, 2223
 - tightUnmarshal, 2223
- activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 2332
 - ~MarshallerFactory, 2333
 - configure, 2333
- activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2411
 - ~MessageAckMarshaller, 2412
 - createObject, 2412
 - getDataStructureType, 2412
 - looseMarshal, 2412
 - looseUnmarshal, 2413
 - MessageAckMarshaller, 2412
 - tightMarshal1, 2413
 - tightMarshal2, 2414
 - tightUnmarshal, 2414
- activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2450
 - ~MessageDispatchMarshaller, 2451
 - createObject, 2451
 - getDataStructureType, 2451
 - looseMarshal, 2451
 - looseUnmarshal, 2452
 - MessageDispatchMarshaller, 2451
 - tightMarshal1, 2452
 - tightMarshal2, 2453
 - tightUnmarshal, 2453
- activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2478
 - ~MessageDispatchNotificationMarshaller, 2480
 - createObject, 2480
 - getDataStructureType, 2480
 - looseUnmarshal, 2480
 - MessageDispatchNotificationMarshaller, 2480
 - tightMarshal1, 2481
 - tightMarshal2, 2481
 - tightUnmarshal, 2482
- activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2503
 - ~MessageIdMarshaller, 2504
 - createObject, 2504
 - getDataStructureType, 2504
 - looseMarshal, 2504
 - looseUnmarshal, 2505
 - MessageIdMarshaller, 2504
 - tightMarshal1, 2505
 - tightMarshal2, 2506
 - tightUnmarshal, 2506

- activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2536
 - ~MessageMarshaller, 2537
 - looseMarshal, 2537
 - looseUnmarshal, 2537
 - MessageMarshaller, 2537
 - tightMarshal1, 2538
 - tightMarshal2, 2538
 - tightUnmarshal, 2539
- activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2580
 - ~MessagePullMarshaller, 2581
 - createObject, 2581
 - getDataStructureType, 2581
 - looseMarshal, 2581
 - looseUnmarshal, 2582
 - MessagePullMarshaller, 2581
 - tightMarshal1, 2582
 - tightMarshal2, 2583
 - tightUnmarshal, 2583
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2632
 - ~NetworkBridgeFilterMarshaller, 2634
 - createObject, 2634
 - getDataStructureType, 2634
 - looseMarshal, 2634
 - looseUnmarshal, 2634
 - NetworkBridgeFilterMarshaller, 2634
 - tightMarshal1, 2635
 - tightMarshal2, 2635
 - tightUnmarshal, 2636
- activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2748
 - ~PartialCommandMarshaller, 2749
 - createObject, 2749
 - getDataStructureType, 2749
 - looseMarshal, 2749
 - looseUnmarshal, 2750
 - PartialCommandMarshaller, 2749
 - tightMarshal1, 2750
 - tightMarshal2, 2751
 - tightUnmarshal, 2751
- activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 2846
 - ~ProducerAckMarshaller, 2848
 - createObject, 2848
 - getDataStructureType, 2848
 - looseMarshal, 2848
 - looseUnmarshal, 2848
 - ProducerAckMarshaller, 2848
 - tightMarshal1, 2849
 - tightMarshal2, 2849
 - tightUnmarshal, 2850
- activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 2878
 - ~ProducerIdMarshaller, 2879
 - createObject, 2879
 - getDataStructureType, 2879
 - looseMarshal, 2879
 - looseUnmarshal, 2880
 - ProducerIdMarshaller, 2879
 - tightMarshal1, 2880
 - tightMarshal2, 2880
 - tightUnmarshal, 2881
- activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 2902
 - ~ProducerInfoMarshaller, 2903
 - createObject, 2903
 - getDataStructureType, 2903
 - looseMarshal, 2904
 - looseUnmarshal, 2904
 - ProducerInfoMarshaller, 2903
 - tightMarshal1, 2904
 - tightMarshal2, 2905
 - tightUnmarshal, 2905
- activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 3011
 - ~RemoveInfoMarshaller, 3012
 - createObject, 3012
 - getDataStructureType, 3012
 - looseMarshal, 3012
 - looseUnmarshal, 3013
 - RemoveInfoMarshaller, 3012
 - tightMarshal1, 3013
 - tightMarshal2, 3014
 - tightUnmarshal, 3014
- activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3039
 - ~RemoveSubscriptionInfoMarshaller, 3040
 - createObject, 3040
 - getDataStructureType, 3040
 - looseMarshal, 3040
 - looseUnmarshal, 3041
 - RemoveSubscriptionInfoMarshaller, 3040
 - tightMarshal1, 3041
 - tightMarshal2, 3042
 - tightUnmarshal, 3042
- activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 3046
 - ~ReplayCommandMarshaller, 3047
 - createObject, 3047
 - getDataStructureType, 3047
 - looseMarshal, 3047
 - looseUnmarshal, 3048
 - ReplayCommandMarshaller, 3047
 - tightMarshal1, 3048
 - tightMarshal2, 3049

- tightUnmarshal, 3049
- activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 3085
 - ~ResponseMarshaller, 3086
 - createObject, 3086
 - getDataStructureType, 3086
 - looseMarshal, 3086
 - looseUnmarshal, 3087
 - ResponseMarshaller, 3086
 - tightMarshal1, 3087
 - tightMarshal2, 3088
 - tightUnmarshal, 3088
- activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 3169
 - ~SessionIdMarshaller, 3170
 - createObject, 3170
 - getDataStructureType, 3170
 - looseMarshal, 3170
 - looseUnmarshal, 3171
 - SessionIdMarshaller, 3170
 - tightMarshal1, 3171
 - tightMarshal2, 3172
 - tightUnmarshal, 3172
- activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 3211
 - ~SessionInfoMarshaller, 3213
 - createObject, 3213
 - getDataStructureType, 3213
 - looseMarshal, 3213
 - looseUnmarshal, 3213
 - SessionInfoMarshaller, 3213
 - tightMarshal1, 3214
 - tightMarshal2, 3214
 - tightUnmarshal, 3215
- activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 3272
 - ~ShutdownInfoMarshaller, 3273
 - createObject, 3273
 - getDataStructureType, 3273
 - looseMarshal, 3273
 - looseUnmarshal, 3274
 - ShutdownInfoMarshaller, 3273
 - tightMarshal1, 3274
 - tightMarshal2, 3275
 - tightUnmarshal, 3275
- activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 3450
 - ~SubscriptionInfoMarshaller, 3451
 - createObject, 3451
 - getDataStructureType, 3451
 - looseMarshal, 3451
 - looseUnmarshal, 3452
 - SubscriptionInfoMarshaller, 3451
 - tightMarshal1, 3452
- tightMarshal2, 3452
- activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3587
 - ~TransactionIdMarshaller, 3588
 - looseMarshal, 3588
 - looseUnmarshal, 3589
 - tightMarshal1, 3589
 - tightMarshal2, 3590
 - tightUnmarshal, 3590
 - TransactionIdMarshaller, 3588
- activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3616
 - ~TransactionInfoMarshaller, 3616
 - createObject, 3616
 - getDataStructureType, 3616
 - looseMarshal, 3616
 - looseUnmarshal, 3616
 - tightMarshal1, 3617
 - tightMarshal2, 3617
 - tightUnmarshal, 3618
 - TransactionInfoMarshaller, 3616
- activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 3741
 - ~WireFormatInfoMarshaller, 3741
 - createObject, 3741
 - getDataStructureType, 3741
 - looseMarshal, 3741
 - looseUnmarshal, 3741
 - tightMarshal1, 3742
 - tightMarshal2, 3742
 - tightUnmarshal, 3743
 - WireFormatInfoMarshaller, 3741
- activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 3778
 - ~XATransactionIdMarshaller, 3778
 - createObject, 3778
 - getDataStructureType, 3778
 - looseMarshal, 3778
 - looseUnmarshal, 3779
 - tightMarshal1, 3779
 - tightMarshal2, 3780
 - tightUnmarshal, 3780
 - XATransactionIdMarshaller, 3778
- activemq::wireformat::openwire::marshal::v5, 188
- activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, 188
 - ~ActiveMQBlobMessageMarshaller, 188
 - ActiveMQBlobMessageMarshaller, 188
 - createObject, 188
 - getDataStructureType, 188
 - looseMarshal, 188
 - looseUnmarshal, 188

- tightMarshal1, 189
- tightMarshal2, 189
- tightUnmarshal, 190
- activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 221
 - ~ActiveMQBytesMessageMarshaller, 223
 - ActiveMQBytesMessageMarshaller, 223
 - createObject, 223
 - getDataStructureType, 223
 - looseMarshal, 223
 - looseUnmarshal, 223
 - tightMarshal1, 224
 - tightMarshal2, 224
 - tightUnmarshal, 225
- activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 301
 - ~ActiveMQDestinationMarshaller, 303
 - ActiveMQDestinationMarshaller, 303
 - looseMarshal, 303
 - looseUnmarshal, 303
 - tightMarshal1, 304
 - tightMarshal2, 304
 - tightUnmarshal, 305
- activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 340
 - ~ActiveMQMapMessageMarshaller, 342
 - ActiveMQMapMessageMarshaller, 342
 - createObject, 342
 - getDataStructureType, 342
 - looseMarshal, 342
 - looseUnmarshal, 342
 - tightMarshal1, 343
 - tightMarshal2, 343
 - tightUnmarshal, 344
- activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 367
 - ~ActiveMQMessageMarshaller, 368
 - ActiveMQMessageMarshaller, 368
 - createObject, 368
 - getDataStructureType, 368
 - looseMarshal, 368
 - looseUnmarshal, 369
 - tightMarshal1, 369
 - tightMarshal2, 370
 - tightUnmarshal, 370
- activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 411
 - ~ActiveMQObjectMessageMarshaller, 412
 - ActiveMQObjectMessageMarshaller, 412
 - createObject, 412
 - getDataStructureType, 412
 - looseMarshal, 412
 - looseUnmarshal, 413
 - tightMarshal1, 413
- tightMarshal2, 414
- tightUnmarshal, 414
- activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, 453
 - ~ActiveMQQueueMessageMarshaller, 454
 - ActiveMQQueueMessageMarshaller, 454
 - createObject, 454
 - getDataStructureType, 454
 - looseMarshal, 454
 - looseUnmarshal, 455
 - tightMarshal1, 455
 - tightMarshal2, 456
 - tightUnmarshal, 456
- activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 511
 - ~ActiveMQStreamMessageMarshaller, 513
 - ActiveMQStreamMessageMarshaller, 513
 - createObject, 513
 - getDataStructureType, 513
 - looseMarshal, 513
 - looseUnmarshal, 513
 - tightMarshal1, 514
 - tightMarshal2, 514
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 538
 - ~ActiveMQTempDestinationMarshaller, 539
 - ActiveMQTempDestinationMarshaller, 539
 - looseMarshal, 539
 - looseUnmarshal, 539
 - tightMarshal1, 540
 - tightMarshal2, 540
 - tightUnmarshal, 541
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 564
 - ~ActiveMQTempQueueMarshaller, 566
 - ActiveMQTempQueueMarshaller, 566
 - createObject, 566
 - getDataStructureType, 566
 - looseMarshal, 566
 - looseUnmarshal, 566
 - tightMarshal1, 567
 - tightMarshal2, 567
 - tightUnmarshal, 568
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 592
 - ~ActiveMQTempTopicMarshaller, 594
 - ActiveMQTempTopicMarshaller, 594
 - createObject, 594
 - getDataStructureType, 594
 - looseMarshal, 594
 - looseUnmarshal, 594
 - tightMarshal1, 595

- tightMarshal2, 595
- tightUnmarshal, 596
- activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 621
 - ~ActiveMQTextMessageMarshaller, 622
 - ActiveMQTextMessageMarshaller, 622
 - createObject, 622
 - getDataStructureType, 622
 - looseMarshal, 622
 - looseUnmarshal, 623
 - tightMarshal1, 623
 - tightMarshal2, 624
 - tightUnmarshal, 624
- activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, 648
 - ~ActiveMQTopicMarshaller, 650
 - ActiveMQTopicMarshaller, 650
 - createObject, 650
 - getDataStructureType, 650
 - looseMarshal, 650
 - looseUnmarshal, 650
 - tightMarshal1, 651
 - tightMarshal2, 651
 - tightUnmarshal, 652
- activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 721
 - ~BaseCommandMarshaller, 722
 - BaseCommandMarshaller, 722
 - looseMarshal, 722
 - looseUnmarshal, 723
 - tightMarshal1, 724
 - tightMarshal2, 725
 - tightUnmarshal, 727
- activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 812
 - ~BrokerIdMarshaller, 814
 - BrokerIdMarshaller, 814
 - createObject, 814
 - getDataStructureType, 814
 - looseMarshal, 814
 - looseUnmarshal, 814
 - tightMarshal1, 815
 - tightMarshal2, 815
 - tightUnmarshal, 816
- activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 843
 - ~BrokerInfoMarshaller, 845
 - BrokerInfoMarshaller, 845
 - createObject, 845
 - getDataStructureType, 845
 - looseMarshal, 845
 - looseUnmarshal, 845
 - tightMarshal1, 846
 - tightMarshal2, 846
- tightUnmarshal, 847
- activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1188
 - ~ConnectionControlMarshaller, 1190
 - ConnectionControlMarshaller, 1190
 - createObject, 1190
 - getDataStructureType, 1190
 - looseMarshal, 1190
 - looseUnmarshal, 1190
 - tightMarshal1, 1191
 - tightMarshal2, 1191
 - tightUnmarshal, 1192
- activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1220
 - ~ConnectionErrorMarshaller, 1221
 - ConnectionErrorMarshaller, 1221
 - createObject, 1221
 - getDataStructureType, 1221
 - looseMarshal, 1221
 - looseUnmarshal, 1222
 - tightMarshal1, 1222
 - tightMarshal2, 1223
 - tightUnmarshal, 1223
- activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1249
 - ~ConnectionIdMarshaller, 1251
 - ConnectionIdMarshaller, 1251
 - createObject, 1251
 - getDataStructureType, 1251
 - looseMarshal, 1251
 - looseUnmarshal, 1251
 - tightMarshal1, 1252
 - tightMarshal2, 1252
 - tightUnmarshal, 1253
- activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1279
 - ~ConnectionInfoMarshaller, 1281
 - ConnectionInfoMarshaller, 1281
 - createObject, 1281
 - getDataStructureType, 1281
 - looseMarshal, 1281
 - looseUnmarshal, 1281
 - tightMarshal1, 1282
 - tightMarshal2, 1282
 - tightUnmarshal, 1283
- activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1322
 - ~ConsumerControlMarshaller, 1324
 - ConsumerControlMarshaller, 1324
 - createObject, 1324
 - getDataStructureType, 1324
 - looseMarshal, 1324
 - looseUnmarshal, 1324
 - tightMarshal1, 1325

- tightMarshal2, 1325
- tightUnmarshal, 1326
- activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1349
 - ~ConsumerInfoMarshaller, 1351
 - ConsumerInfoMarshaller, 1351
 - createObject, 1351
 - getDataStructureType, 1351
 - looseMarshal, 1351
 - looseUnmarshal, 1351
 - tightMarshal1, 1352
 - tightMarshal2, 1352
 - tightUnmarshal, 1353
- activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1381
 - ~ConsumerInfoMarshaller, 1383
 - ConsumerInfoMarshaller, 1383
 - createObject, 1383
 - getDataStructureType, 1383
 - looseMarshal, 1383
 - looseUnmarshal, 1383
 - tightMarshal1, 1384
 - tightMarshal2, 1384
 - tightUnmarshal, 1385
- activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1409
 - ~ControlCommandMarshaller, 1410
 - ControlCommandMarshaller, 1410
 - createObject, 1410
 - getDataStructureType, 1410
 - looseMarshal, 1410
 - looseUnmarshal, 1411
 - tightMarshal1, 1411
 - tightMarshal2, 1412
 - tightUnmarshal, 1412
- activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1442
 - ~DataArrayResponseMarshaller, 1443
 - createObject, 1443
 - DataArrayResponseMarshaller, 1443
 - getDataStructureType, 1443
 - looseMarshal, 1443
 - looseUnmarshal, 1444
 - tightMarshal1, 1444
 - tightMarshal2, 1445
 - tightUnmarshal, 1445
- activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1479
 - ~DataResponseMarshaller, 1481
 - createObject, 1481
 - DataResponseMarshaller, 1481
 - getDataStructureType, 1481
 - looseMarshal, 1481
 - looseUnmarshal, 1482
- tightMarshal1, 1482
- tightMarshal2, 1482
- tightMarshal, 1483
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1638
 - ~DestinationInfoMarshaller, 1639
 - createObject, 1639
 - DestinationInfoMarshaller, 1639
 - getDataStructureType, 1639
 - looseMarshal, 1639
 - looseUnmarshal, 1640
 - tightMarshal1, 1640
 - tightMarshal2, 1641
- activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1666
 - ~DiscoveryEventMarshaller, 1668
 - createObject, 1668
 - DiscoveryEventMarshaller, 1668
 - getDataStructureType, 1668
 - looseMarshal, 1668
 - looseUnmarshal, 1668
 - tightMarshal1, 1669
 - tightMarshal2, 1669
- activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1734
 - ~ExceptionResponseMarshaller, 1736
 - createObject, 1736
 - ExceptionResponseMarshaller, 1736
 - getDataStructureType, 1736
 - looseMarshal, 1736
 - looseUnmarshal, 1737
 - tightMarshal1, 1737
 - tightMarshal2, 1737
- activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 1834
 - ~FlushCommandMarshaller, 1836
 - createObject, 1836
 - FlushCommandMarshaller, 1836
 - getDataStructureType, 1836
 - looseMarshal, 1836
 - looseUnmarshal, 1836
 - tightMarshal1, 1837
 - tightMarshal2, 1837
- activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 1978
 - ~IntegerResponseMarshaller, 1980
 - createObject, 1980
 - getDataStructureType, 1980
 - IntegerResponseMarshaller, 1980
 - looseMarshal, 1980

- looseUnmarshal, 1981
- tightMarshal1, 1981
- tightMarshal2, 1981
- tightUnmarshal, 1982
- activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 2025
 - ~JournalQueueAckMarshaller, 2026
 - createObject, 2026
 - getDataStructureType, 2026
 - JournalQueueAckMarshaller, 2026
 - looseMarshal, 2026
 - looseUnmarshal, 2027
 - tightMarshal1, 2027
 - tightMarshal2, 2027
 - tightUnmarshal, 2028
- activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 2045
 - ~JournalTopicAckMarshaller, 2047
 - createObject, 2047
 - getDataStructureType, 2047
 - JournalTopicAckMarshaller, 2047
 - looseMarshal, 2047
 - looseUnmarshal, 2047
 - tightMarshal1, 2048
 - tightMarshal2, 2048
 - tightUnmarshal, 2049
- activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 2091
 - ~JournalTraceMarshaller, 2093
 - createObject, 2093
 - getDataStructureType, 2093
 - JournalTraceMarshaller, 2093
 - looseMarshal, 2093
 - looseUnmarshal, 2093
 - tightMarshal1, 2094
 - tightMarshal2, 2094
 - tightUnmarshal, 2095
- activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 2110
 - ~JournalTransactionMarshaller, 2112
 - createObject, 2112
 - getDataStructureType, 2112
 - JournalTransactionMarshaller, 2112
 - looseMarshal, 2112
 - looseUnmarshal, 2112
 - tightMarshal1, 2113
 - tightMarshal2, 2113
 - tightUnmarshal, 2114
- activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 2141
 - ~KeepAliveInfoMarshaller, 2142
 - createObject, 2142
 - getDataStructureType, 2142
 - KeepAliveInfoMarshaller, 2142
 - looseMarshal, 2142
 - looseUnmarshal, 2143
 - tightMarshal1, 2143
 - tightMarshal2, 2144
 - tightUnmarshal, 2144
- activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 2170
 - ~LastPartialCommandMarshaller, 2171
 - createObject, 2171
 - getDataStructureType, 2171
 - LastPartialCommandMarshaller, 2171
 - looseMarshal, 2172
 - looseUnmarshal, 2172
 - tightMarshal1, 2172
 - tightMarshal2, 2173
 - tightUnmarshal, 2173
- activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2216
 - ~LocalTransactionIdMarshaller, 2217
 - createObject, 2217
 - getDataStructureType, 2217
 - LocalTransactionIdMarshaller, 2217
 - looseMarshal, 2217
 - looseUnmarshal, 2218
 - tightMarshal1, 2218
 - tightMarshal2, 2219
 - tightUnmarshal, 2219
- activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2333
 - ~MarshallerFactory, 2333
 - configure, 2333
- activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2419
 - ~MessageAckMarshaller, 2420
 - createObject, 2420
 - getDataStructureType, 2420
 - looseMarshal, 2420
 - looseUnmarshal, 2421
 - MessageAckMarshaller, 2420
 - tightMarshal1, 2421
 - tightMarshal2, 2422
 - tightUnmarshal, 2422
- activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2446
 - ~MessageDispatchMarshaller, 2447
 - createObject, 2447
 - getDataStructureType, 2447
 - InfoMarshaller, 2447
 - looseUnmarshal, 2448
 - MessageDispatchMarshaller, 2447
 - tightMarshal1, 2448
 - tightMarshal2, 2449
 - tightUnmarshal, 2449

- activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2486
 - ~MessageDispatchNotificationMarshaller, 2488
 - createObject, 2488
 - getDataStructureType, 2488
 - looseMarshal, 2488
 - looseUnmarshal, 2488
 - MessageDispatchNotificationMarshaller, 2488
 - tightMarshal1, 2489
 - tightMarshal2, 2489
 - tightUnmarshal, 2490
- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2507
 - ~MessageIdMarshaller, 2508
 - createObject, 2508
 - getDataStructureType, 2508
 - looseMarshal, 2508
 - looseUnmarshal, 2509
 - MessageIdMarshaller, 2508
 - tightMarshal1, 2509
 - tightMarshal2, 2509
 - tightUnmarshal, 2510
- activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2523
 - ~MessageMarshaller, 2524
 - looseMarshal, 2524
 - looseUnmarshal, 2525
 - MessageMarshaller, 2524
 - tightMarshal1, 2525
 - tightMarshal2, 2526
 - tightUnmarshal, 2527
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2572
 - ~MessagePullMarshaller, 2573
 - createObject, 2573
 - getDataStructureType, 2573
 - looseMarshal, 2573
 - looseUnmarshal, 2574
 - MessagePullMarshaller, 2573
 - tightMarshal1, 2574
 - tightMarshal2, 2575
 - tightUnmarshal, 2575
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2624
 - ~NetworkBridgeFilterMarshaller, 2626
 - createObject, 2626
 - getDataStructureType, 2626
 - looseMarshal, 2626
 - looseUnmarshal, 2626
 - NetworkBridgeFilterMarshaller, 2626
 - tightMarshal1, 2627
 - tightMarshal2, 2627
- activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2739
 - ~PartialCommandMarshaller, 2741
 - createObject, 2741
 - getDataStructureType, 2741
 - looseMarshal, 2741
 - looseUnmarshal, 2742
 - PartialCommandMarshaller, 2741
 - tightMarshal1, 2742
 - tightMarshal2, 2742
 - tightUnmarshal, 2743
- activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 2856
 - ~ProducerAckMarshaller, 2856
 - createObject, 2856
 - getDataStructureType, 2856
 - looseMarshal, 2856
 - looseUnmarshal, 2856
 - ProducerAckMarshaller, 2856
 - tightMarshal1, 2857
 - tightMarshal2, 2857
 - tightUnmarshal, 2858
- activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 2887
 - ~ProducerIdMarshaller, 2887
 - createObject, 2887
 - getDataStructureType, 2887
 - looseMarshal, 2887
 - looseUnmarshal, 2887
 - ProducerIdMarshaller, 2887
 - tightMarshal1, 2888
 - tightMarshal2, 2888
- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2914
 - ~ProducerInfoMarshaller, 2915
 - createObject, 2915
 - getDataStructureType, 2915
 - looseMarshal, 2915
 - looseUnmarshal, 2916
 - ProducerInfoMarshaller, 2915
 - tightMarshal1, 2916
 - tightMarshal2, 2917
- activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 3007
 - ~RemoveInfoMarshaller, 3008
 - createObject, 3008
 - getDataStructureType, 3008
 - looseMarshal, 3008
 - looseUnmarshal, 3009
 - RemoveInfoMarshaller, 3008
 - tightMarshal1, 3009

- tightMarshal2, 3010
- tightUnmarshal, 3010
- activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3035
 - ~RemoveSubscriptionInfoMarshaller, 3036
 - createObject, 3036
 - getDataStructureType, 3036
 - looseMarshal, 3036
 - looseUnmarshal, 3037
 - RemoveSubscriptionInfoMarshaller, 3036
 - tightMarshal1, 3037
 - tightMarshal2, 3038
 - tightUnmarshal, 3038
- activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 3066
 - ~ReplayCommandMarshaller, 3067
 - createObject, 3067
 - getDataStructureType, 3067
 - looseMarshal, 3067
 - looseUnmarshal, 3068
 - ReplayCommandMarshaller, 3067
 - tightMarshal1, 3068
 - tightMarshal2, 3069
 - tightUnmarshal, 3069
- activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 3093
 - ~ResponseMarshaller, 3095
 - createObject, 3095
 - getDataStructureType, 3095
 - looseMarshal, 3095
 - looseUnmarshal, 3096
 - ResponseMarshaller, 3095
 - tightMarshal1, 3096
 - tightMarshal2, 3097
 - tightUnmarshal, 3097
- activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 3176
 - ~SessionIdMarshaller, 3178
 - createObject, 3178
 - getDataStructureType, 3178
 - looseMarshal, 3178
 - looseUnmarshal, 3178
 - SessionIdMarshaller, 3178
 - tightMarshal1, 3179
 - tightMarshal2, 3179
 - tightUnmarshal, 3180
- activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 3195
 - ~SessionInfoMarshaller, 3197
 - createObject, 3197
 - getDataStructureType, 3197
 - looseMarshal, 3197
 - looseUnmarshal, 3197
 - SessionInfoMarshaller, 3197
- tightMarshal1, 3198
- tightMarshal2, 3198
- activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 3264
 - ~ShutdownInfoMarshaller, 3265
 - createObject, 3265
 - getDataStructureType, 3265
 - looseMarshal, 3265
 - looseUnmarshal, 3266
 - ShutdownInfoMarshaller, 3265
 - tightMarshal1, 3266
 - tightMarshal2, 3267
- activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 3446
 - ~SubscriptionInfoMarshaller, 3447
 - createObject, 3447
 - getDataStructureType, 3447
 - looseMarshal, 3447
 - looseUnmarshal, 3448
 - SubscriptionInfoMarshaller, 3447
 - tightMarshal1, 3448
 - tightMarshal2, 3449
- activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3572
 - ~TransactionIdMarshaller, 3574
 - looseMarshal, 3574
 - looseUnmarshal, 3574
 - tightMarshal1, 3575
 - tightMarshal2, 3575
 - tightUnmarshal, 3576
 - TransactionIdMarshaller, 3574
- activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3600
 - ~TransactionInfoMarshaller, 3600
 - createObject, 3600
 - getDataStructureType, 3600
 - looseMarshal, 3600
 - looseUnmarshal, 3600
 - tightMarshal1, 3601
 - tightMarshal2, 3601
 - tightUnmarshal, 3602
 - TransactionInfoMarshaller, 3600
- activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3729
 - ~WireFormatInfoMarshaller, 3729
 - createObject, 3729
 - getDataStructureType, 3729
 - looseMarshal, 3729
 - looseUnmarshal, 3730
 - tightMarshal1, 3730
 - tightMarshal2, 3730

- tightUnmarshal, 3731
- WireFormatInfoMarshaller, 3729
- activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 3789
 - ~XATransactionIdMarshaller, 3790
 - createObject, 3790
 - getDataStructureType, 3790
 - looseMarshal, 3790
 - looseUnmarshal, 3791
 - tightMarshal1, 3791
 - tightMarshal2, 3792
 - tightUnmarshal, 3792
 - XATransactionIdMarshaller, 3790
- activemq::wireformat::openwire::marshal::v6, 113
- activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller, 190
 - ~ActiveMQBlobMessageMarshaller, 192
 - ActiveMQBlobMessageMarshaller, 192
 - createObject, 192
 - getDataStructureType, 192
 - looseMarshal, 192
 - looseUnmarshal, 192
 - tightMarshal1, 193
 - tightMarshal2, 193
 - tightUnmarshal, 194
- activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller, 225
 - ~ActiveMQBytesMessageMarshaller, 227
 - ActiveMQBytesMessageMarshaller, 227
 - createObject, 227
 - getDataStructureType, 227
 - looseMarshal, 227
 - looseUnmarshal, 227
 - tightMarshal1, 228
 - tightMarshal2, 228
 - tightUnmarshal, 229
- activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller, 309
 - ~ActiveMQDestinationMarshaller, 311
 - ActiveMQDestinationMarshaller, 311
 - looseMarshal, 311
 - looseUnmarshal, 311
 - tightMarshal1, 312
 - tightMarshal2, 312
 - tightUnmarshal, 313
- activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller, 348
 - ~ActiveMQMapMessageMarshaller, 350
 - ActiveMQMapMessageMarshaller, 350
 - createObject, 350
 - getDataStructureType, 350
 - looseMarshal, 350
 - looseUnmarshal, 350
- tightMarshal1, 351
- tightMarshal2, 351
- tightUnmarshal, 352
- activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller, 375
 - ~ActiveMQMessageMarshaller, 376
 - ActiveMQMessageMarshaller, 376
 - createObject, 376
 - getDataStructureType, 376
 - looseMarshal, 376
 - looseUnmarshal, 377
 - tightMarshal1, 377
 - tightMarshal2, 378
 - tightUnmarshal, 378
- activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller, 420
 - ~ActiveMQObjectMessageMarshaller, 420
 - ActiveMQObjectMessageMarshaller, 420
 - createObject, 420
 - getDataStructureType, 420
 - looseMarshal, 420
 - looseUnmarshal, 421
 - tightMarshal1, 421
 - tightMarshal2, 422
 - tightUnmarshal, 422
- activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMessageMarshaller, 462
 - ~ActiveMQQueueMarshaller, 462
 - ActiveMQQueueMarshaller, 462
 - createObject, 462
 - getDataStructureType, 462
 - looseMarshal, 462
 - looseUnmarshal, 463
 - tightMarshal1, 463
 - tightMarshal2, 464
 - tightUnmarshal, 464
- activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller, 519
 - ~ActiveMQStreamMessageMarshaller, 521
 - ActiveMQStreamMessageMarshaller, 521
 - createObject, 521
 - getDataStructureType, 521
 - looseMarshal, 521
 - looseUnmarshal, 521
 - tightMarshal1, 522
 - tightMarshal2, 522
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller, 545
 - ~ActiveMQTempDestinationMarshaller, 546
 - ActiveMQTempDestinationMarshaller, 546
 - looseMarshal, 546
 - looseUnmarshal, 546

- tightMarshal1, 547
- tightMarshal2, 547
- tightUnmarshal, 548
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller, 572
 - ~ActiveMQTempQueueMarshaller, 574
 - ActiveMQTempQueueMarshaller, 574
 - createObject, 574
 - getDataStructureType, 574
 - looseMarshal, 574
 - looseUnmarshal, 574
 - tightMarshal1, 575
 - tightMarshal2, 575
 - tightUnmarshal, 576
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller, 600
 - ~ActiveMQTempTopicMarshaller, 602
 - ActiveMQTempTopicMarshaller, 602
 - createObject, 602
 - getDataStructureType, 602
 - looseMarshal, 602
 - looseUnmarshal, 602
 - tightMarshal1, 603
 - tightMarshal2, 603
 - tightUnmarshal, 604
- activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller, 625
 - ~ActiveMQTextMessageMarshaller, 626
 - ActiveMQTextMessageMarshaller, 626
 - createObject, 626
 - getDataStructureType, 626
 - looseMarshal, 626
 - looseUnmarshal, 627
 - tightMarshal1, 627
 - tightMarshal2, 628
 - tightUnmarshal, 628
- activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller, 652
 - ~ActiveMQTopicMarshaller, 654
 - ActiveMQTopicMarshaller, 654
 - createObject, 654
 - getDataStructureType, 654
 - looseMarshal, 654
 - looseUnmarshal, 654
 - tightMarshal1, 655
 - tightMarshal2, 655
 - tightUnmarshal, 656
- activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller, 728
 - ~BaseCommandMarshaller, 729
 - BaseCommandMarshaller, 729
 - looseMarshal, 729
 - looseUnmarshal, 730
 - tightMarshal1, 731
 - tightMarshal2, 732
 - tightUnmarshal, 733
- activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller, 816
 - ~BrokerIdMarshaller, 818
 - BrokerIdMarshaller, 818
 - createObject, 818
 - getDataStructureType, 818
 - looseMarshal, 818
 - looseUnmarshal, 818
 - tightMarshal1, 819
 - tightMarshal2, 819
 - tightUnmarshal, 820
- activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller, 847
 - ~BrokerInfoMarshaller, 849
 - BrokerInfoMarshaller, 849
 - createObject, 849
 - getDataStructureType, 849
 - looseMarshal, 849
 - looseUnmarshal, 849
 - tightMarshal1, 850
 - tightMarshal2, 850
 - tightUnmarshal, 851
- activemq::wireformat::openwire::marshal::v6::ConnectionControlMessageMarshaller, 1192
 - ~ConnectionControlMarshaller, 1194
 - ConnectionControlMarshaller, 1194
 - createObject, 1194
 - getDataStructureType, 1194
 - looseMarshal, 1194
 - looseUnmarshal, 1194
 - tightMarshal1, 1195
 - tightMarshal2, 1195
 - tightUnmarshal, 1196
- activemq::wireformat::openwire::marshal::v6::ConnectionErrorMessageMarshaller, 1224
 - ~ConnectionErrorMessageMarshaller, 1225
 - ConnectionErrorMessageMarshaller, 1225
 - createObject, 1225
 - getDataStructureType, 1225
 - looseMarshal, 1225
 - looseUnmarshal, 1226
 - tightMarshal1, 1226
 - tightMarshal2, 1227
 - tightUnmarshal, 1227
- activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller, 1254
 - ~ConnectionIdMarshaller, 1255
 - ConnectionIdMarshaller, 1255
 - createObject, 1255
 - getDataStructureType, 1255
 - looseMarshal, 1255
 - looseUnmarshal, 1255

- tightMarshal1, 1256
- tightMarshal2, 1256
- tightUnmarshal, 1257
- activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller, 1283
 - ~ConnectionInfoMarshaller, 1285
 - ConnectionInfoMarshaller, 1285
 - createObject, 1285
 - getDataStructureType, 1285
 - looseMarshal, 1285
 - looseUnmarshal, 1285
 - tightMarshal1, 1286
 - tightMarshal2, 1286
 - tightUnmarshal, 1287
- activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller, 1326
 - ~ConsumerControlMarshaller, 1328
 - ConsumerControlMarshaller, 1328
 - createObject, 1328
 - getDataStructureType, 1328
 - looseMarshal, 1328
 - looseUnmarshal, 1328
 - tightMarshal1, 1329
 - tightMarshal2, 1329
 - tightUnmarshal, 1330
- activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 1353
 - ~ConsumerIdMarshaller, 1355
 - ConsumerIdMarshaller, 1355
 - createObject, 1355
 - getDataStructureType, 1355
 - looseMarshal, 1355
 - looseUnmarshal, 1355
 - tightMarshal1, 1356
 - tightMarshal2, 1356
 - tightUnmarshal, 1357
- activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1385
 - ~ConsumerInfoMarshaller, 1387
 - ConsumerInfoMarshaller, 1387
 - createObject, 1387
 - getDataStructureType, 1387
 - looseMarshal, 1387
 - looseUnmarshal, 1387
 - tightMarshal1, 1388
 - tightMarshal2, 1388
 - tightUnmarshal, 1389
- activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller, 1413
 - ~ControlCommandMarshaller, 1414
 - ControlCommandMarshaller, 1414
 - createObject, 1414
 - getDataStructureType, 1414
 - looseMarshal, 1414
 - looseUnmarshal, 1415
 - tightMarshal1, 1415
 - tightMarshal2, 1416
 - tightUnmarshal, 1416
- activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller, 1446
 - ~DataArrayResponseMarshaller, 1447
 - createObject, 1447
 - DataArrayResponseMarshaller, 1447
 - getDataStructureType, 1447
 - looseMarshal, 1447
 - looseUnmarshal, 1448
 - tightMarshal1, 1448
 - tightMarshal2, 1449
- activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 1483
 - ~DataResponseMarshaller, 1485
 - createObject, 1485
 - DataResponseMarshaller, 1485
 - getDataStructureType, 1485
 - looseMarshal, 1485
 - looseUnmarshal, 1486
 - tightMarshal1, 1486
 - tightMarshal2, 1486
- activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1634
 - ~DestinationInfoMarshaller, 1635
 - createObject, 1635
 - DestinationInfoMarshaller, 1635
 - getDataStructureType, 1635
 - looseMarshal, 1635
 - looseUnmarshal, 1636
 - tightMarshal1, 1636
 - tightMarshal2, 1637
- activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1647
 - ~DiscoveryEventMarshaller, 1648
 - createObject, 1648
 - DiscoveryEventMarshaller, 1648
 - getDataStructureType, 1648
 - looseMarshal, 1648
 - looseUnmarshal, 1649
 - tightMarshal1, 1649
 - tightMarshal2, 1650
- activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 1722
 - ~ExceptionResponseMarshaller, 1724
 - createObject, 1724
 - ExceptionResponseMarshaller, 1724
 - getDataStructureType, 1724

- looseMarshal, 1724
- looseUnmarshal, 1725
- tightMarshal1, 1725
- tightMarshal2, 1725
- tightUnmarshal, 1726
- activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, 1814
 - ~FlushCommandMarshaller, 1816
 - createObject, 1816
 - FlushCommandMarshaller, 1816
 - getDataStructureType, 1816
 - looseMarshal, 1816
 - looseUnmarshal, 1816
 - tightMarshal1, 1817
 - tightMarshal2, 1817
 - tightUnmarshal, 1818
- activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 1958
 - ~IntegerResponseMarshaller, 1960
 - createObject, 1960
 - getDataStructureType, 1960
 - IntegerResponseMarshaller, 1960
 - looseMarshal, 1960
 - looseUnmarshal, 1961
 - tightMarshal1, 1961
 - tightMarshal2, 1961
 - tightUnmarshal, 1962
- activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, 2017
 - ~JournalQueueAckMarshaller, 2018
 - createObject, 2018
 - getDataStructureType, 2018
 - JournalQueueAckMarshaller, 2018
 - looseMarshal, 2019
 - looseUnmarshal, 2019
 - tightMarshal1, 2019
 - tightMarshal2, 2020
 - tightUnmarshal, 2020
- activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, 2057
 - ~JournalTopicAckMarshaller, 2059
 - createObject, 2059
 - getDataStructureType, 2059
 - JournalTopicAckMarshaller, 2059
 - looseMarshal, 2059
 - looseUnmarshal, 2059
 - tightMarshal1, 2060
 - tightMarshal2, 2060
 - tightUnmarshal, 2061
- activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller, 2079
 - ~JournalTraceMarshaller, 2081
 - createObject, 2081
 - getDataStructureType, 2081
 - JournalTraceMarshaller, 2081
 - looseMarshal, 2081
 - looseUnmarshal, 2081
 - tightMarshal1, 2082
 - tightMarshal2, 2082
 - tightUnmarshal, 2083
- activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, 2099
 - ~JournalTransactionMarshaller, 2100
 - createObject, 2100
 - getDataStructureType, 2100
 - JournalTransactionMarshaller, 2100
 - looseMarshal, 2100
 - looseUnmarshal, 2101
 - tightMarshal1, 2101
 - tightMarshal2, 2101
- activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2125
 - ~KeepAliveInfoMarshaller, 2126
 - createObject, 2126
 - getDataStructureType, 2126
 - KeepAliveInfoMarshaller, 2126
 - looseMarshal, 2127
 - looseUnmarshal, 2127
 - tightMarshal1, 2127
 - tightMarshal2, 2128
- activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, 2158
 - ~LastPartialCommandMarshaller, 2159
 - createObject, 2159
 - getDataStructureType, 2160
 - LastPartialCommandMarshaller, 2159
 - looseMarshal, 2160
 - looseUnmarshal, 2160
 - tightMarshal1, 2161
 - tightMarshal2, 2161
- activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2204
 - ~LocalTransactionIdMarshaller, 2205
 - createObject, 2205
 - getDataStructureType, 2205
 - LocalTransactionIdMarshaller, 2205
 - looseMarshal, 2205
 - looseUnmarshal, 2206
 - tightMarshal1, 2206
 - tightMarshal2, 2207
- activemq::wireformat::openwire::marshal::v6::MarshallerFactory, 2331
 - ~MarshallerFactory, 2331
 - configure, 2331

- activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2547
- 2399
- ~MessageAckMarshaller, 2400
- createObject, 2400
- getDataStructureType, 2401
- looseMarshal, 2401
- looseUnmarshal, 2401
- MessageAckMarshaller, 2400
- tightMarshal1, 2401
- tightMarshal2, 2402
- tightUnmarshal, 2402
- activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2591
- 2458
- ~MessageDispatchMarshaller, 2459
- createObject, 2459
- getDataStructureType, 2459
- looseMarshal, 2459
- looseUnmarshal, 2460
- MessageDispatchMarshaller, 2459
- tightMarshal1, 2460
- tightMarshal2, 2461
- tightUnmarshal, 2461
- activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2624
- 2466
- ~MessageDispatchNotificationMarshaller, 2468
- createObject, 2468
- getDataStructureType, 2468
- looseMarshal, 2468
- looseUnmarshal, 2468
- MessageDispatchNotificationMarshaller, 2468
- tightMarshal1, 2469
- tightMarshal2, 2469
- tightUnmarshal, 2470
- activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 2514
- 2514
- ~MessageIdMarshaller, 2516
- createObject, 2516
- getDataStructureType, 2516
- looseMarshal, 2516
- looseUnmarshal, 2516
- MessageIdMarshaller, 2516
- tightMarshal1, 2517
- tightMarshal2, 2517
- tightUnmarshal, 2518
- activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2544
- 2544
- ~MessageMarshaller, 2545
- looseMarshal, 2545
- looseUnmarshal, 2545
- MessageMarshaller, 2545
- tightMarshal1, 2546
- tightMarshal2, 2546
- activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2588
- 2588
- ~MessagePullMarshaller, 2589
- createObject, 2589
- getDataStructureType, 2589
- looseMarshal, 2589
- looseUnmarshal, 2590
- MessagePullMarshaller, 2589
- tightMarshal1, 2590
- tightMarshal2, 2591
- activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2622
- 2620
- ~NetworkBridgeFilterMarshaller, 2622
- createObject, 2622
- getDataStructureType, 2622
- looseMarshal, 2622
- looseUnmarshal, 2622
- NetworkBridgeFilterMarshaller, 2622
- tightMarshal1, 2623
- tightMarshal2, 2623
- activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2731
- 2731
- ~PartialCommandMarshaller, 2732
- createObject, 2732
- getDataStructureType, 2732
- looseMarshal, 2733
- looseUnmarshal, 2733
- PartialCommandMarshaller, 2732
- tightMarshal1, 2733
- tightMarshal2, 2734
- tightUnmarshal, 2734
- activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 2860
- 2858
- ~ProducerAckMarshaller, 2860
- createObject, 2860
- getDataStructureType, 2860
- looseMarshal, 2860
- looseUnmarshal, 2860
- ProducerAckMarshaller, 2860
- tightMarshal1, 2861
- tightMarshal2, 2861
- tightUnmarshal, 2862
- activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 2891
- 2889
- ~ProducerIdMarshaller, 2891
- createObject, 2891
- getDataStructureType, 2891
- looseMarshal, 2891
- looseUnmarshal, 2891
- ProducerIdMarshaller, 2891
- tightMarshal1, 2892

- tightMarshal2, 2892
- tightUnmarshal, 2893
- activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 2922
 - ~ProducerInfoMarshaller, 2923
 - createObject, 2923
 - getDataStructureType, 2923
 - looseMarshal, 2923
 - looseUnmarshal, 2924
 - ProducerInfoMarshaller, 2923
 - tightMarshal1, 2924
 - tightMarshal2, 2925
 - tightUnmarshal, 2925
- activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 2995
 - ~RemoveInfoMarshaller, 2996
 - createObject, 2996
 - getDataStructureType, 2996
 - looseMarshal, 2996
 - looseUnmarshal, 2997
 - RemoveInfoMarshaller, 2996
 - tightMarshal1, 2997
 - tightMarshal2, 2998
 - tightUnmarshal, 2998
- activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3031
 - ~RemoveSubscriptionInfoMarshaller, 3032
 - createObject, 3032
 - getDataStructureType, 3032
 - looseMarshal, 3032
 - looseUnmarshal, 3033
 - RemoveSubscriptionInfoMarshaller, 3032
 - tightMarshal1, 3033
 - tightMarshal2, 3034
 - tightUnmarshal, 3034
- activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3062
 - ~ReplayCommandMarshaller, 3063
 - createObject, 3063
 - getDataStructureType, 3063
 - looseMarshal, 3063
 - looseUnmarshal, 3064
 - ReplayCommandMarshaller, 3063
 - tightMarshal1, 3064
 - tightMarshal2, 3065
 - tightUnmarshal, 3065
- activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3107
 - ~ResponseMarshaller, 3108
 - createObject, 3108
 - getDataStructureType, 3108
 - looseMarshal, 3109
 - looseUnmarshal, 3109
 - ResponseMarshaller, 3108
- tightMarshal1, 3110
- tightMarshal2, 3110
- activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3173
 - ~SessionIdMarshaller, 3174
 - createObject, 3174
 - getDataStructureType, 3174
 - looseMarshal, 3174
 - looseUnmarshal, 3175
 - SessionIdMarshaller, 3174
 - tightMarshal1, 3175
 - tightMarshal2, 3175
- activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3192
 - ~SessionInfoMarshaller, 3193
 - createObject, 3193
 - getDataStructureType, 3193
 - looseMarshal, 3193
 - looseUnmarshal, 3193
 - SessionInfoMarshaller, 3193
 - tightMarshal1, 3194
 - tightMarshal2, 3194
- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3252
 - ~ShutdownInfoMarshaller, 3253
 - createObject, 3253
 - getDataStructureType, 3253
 - looseMarshal, 3254
 - looseUnmarshal, 3254
 - ShutdownInfoMarshaller, 3253
 - tightMarshal1, 3254
 - tightMarshal2, 3255
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3453
 - ~SubscriptionInfoMarshaller, 3455
 - createObject, 3455
 - getDataStructureType, 3455
 - looseMarshal, 3455
 - looseUnmarshal, 3455
 - SubscriptionInfoMarshaller, 3455
 - tightMarshal1, 3456
 - tightMarshal2, 3456
- activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3591
 - ~TransactionIdMarshaller, 3592
 - looseMarshal, 3592
 - looseUnmarshal, 3592
 - tightMarshal1, 3593
 - tightMarshal2, 3593

- tightUnmarshal, 3594
- TransactionIdMarshaller, 3592
- activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3610
- ~TransactionInfoMarshaller, 3612
- createObject, 3612
- getDataStructureType, 3612
- looseMarshal, 3612
- looseUnmarshal, 3612
- tightMarshal1, 3613
- tightMarshal2, 3613
- tightUnmarshal, 3614
- TransactionInfoMarshaller, 3612
- activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 3731
- ~WireFormatInfoMarshaller, 3733
- createObject, 3733
- getDataStructureType, 3733
- looseMarshal, 3733
- looseUnmarshal, 3733
- tightMarshal1, 3734
- tightMarshal2, 3734
- tightUnmarshal, 3735
- WireFormatInfoMarshaller, 3733
- activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 3769
- ~XATransactionIdMarshaller, 3770
- createObject, 3770
- getDataStructureType, 3770
- looseMarshal, 3771
- looseUnmarshal, 3771
- tightMarshal1, 3771
- tightMarshal2, 3772
- tightUnmarshal, 3772
- XATransactionIdMarshaller, 3770
- activemq::wireformat::openwire::OpenWireFormat, 2700
- ~OpenWireFormat, 2703
- addMarshaller, 2703
- createNegotiator, 2703
- DEFAULT_VERSION, 2712
- destroyMarshalers, 2704
- doUnmarshal, 2704
- getCacheSize, 2704
- getMaxInactivityDuration, 2704
- getMaxInactivityDurationInitialDelay, 2704
- getPreferredWireFormatInfo, 2705
- getVersion, 2705
- hasNegotiator, 2705
- inReceive, 2705
- isCacheEnabled, 2705
- isSizePrefixDisabled, 2706
- isStackTraceEnabled, 2706
- isTcpNoDelayEnabled, 2706
- isTightEncodingEnabled, 2706
- looseMarshalNestedObject, 2706
- looseUnmarshalNestedObject, 2707
- marshal, 2707
- NULL_TYPE, 2712
- OpenWireFormat, 2703
- renegotiateWireFormat, 2708
- setCacheEnabled, 2708
- setCacheSize, 2708
- setMaxInactivityDuration, 2708
- setMaxInactivityDurationInitialDelay, 2708
- setPrefixDisabled, 2709
- setSizePrefixDisabled, 2709
- setStackTraceEnabled, 2709
- setTcpNoDelayEnabled, 2709
- setTightEncodingEnabled, 2709
- setVersion, 2710
- tightMarshalNestedObject1, 2710
- tightMarshalNestedObject2, 2710
- tightUnmarshalNestedObject, 2711
- unmarshal, 2711
- activemq::wireformat::openwire::OpenWireFormatFactory, 2712
- ~OpenWireFormatFactory, 2712
- createWireFormat, 2713
- OpenWireFormatFactory, 2712
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2713
- ~OpenWireFormatNegotiator, 2714
- close, 2714
- onCommand, 2714
- oneway, 2715
- onTransportException, 2715
- OpenWireFormatNegotiator, 2714
- request, 2715, 2716
- start, 2716
- activemq::wireformat::openwire::OpenWireResponseBuilder, 2717
- ~OpenWireResponseBuilder, 2718
- buildIncomingCommands, 2718
- buildResponse, 2718
- OpenWireResponseBuilder, 2718
- activemq::wireformat::openwire::utils, 117
- activemq::wireformat::openwire::utils::BooleanStream, 787
- ~BooleanStream, 789
- BooleanStream, 789
- clear, 789
- marshal, 789
- marshalledSize, 789
- readBoolean, 789
- unmarshal, 789

- writeBoolean, 790
- activemq::wireformat::openwire::utils::HexTable, 1857
 - ~HexTable, 1858
 - HexTable, 1858
 - operator[], 1858
 - size, 1858
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2557
 - ~MessagePropertyInterceptor, 2559
 - getBooleanProperty, 2559
 - getByteProperty, 2559
 - getDoubleProperty, 2560
 - getFloatProperty, 2560
 - getIntProperty, 2560
 - getLongProperty, 2560
 - getShortProperty, 2561
 - getStringProperty, 2561
 - MessagePropertyInterceptor, 2559
 - setBooleanProperty, 2561
 - setByteProperty, 2561
 - setDoubleProperty, 2562
 - setFloatProperty, 2562
 - setIntProperty, 2562
 - setLongProperty, 2562
 - setShortProperty, 2563
 - setStringProperty, 2563
- activemq::wireformat::stomp, 117
- activemq::wireformat::stomp::StompCommandConstants, 3395
 - ABORT, 3397
 - ACK, 3397
 - ACK_AUTO, 3397
 - ACK_CLIENT, 3397
 - ACK_INDIVIDUAL, 3397
 - BEGIN, 3397
 - BYTES, 3397
 - COMMIT, 3397
 - CONNECT, 3397
 - CONNECTED, 3397
 - DISCONNECT, 3397
 - ERROR_CMD, 3397
 - HEADER_ACK, 3397
 - HEADER_CLIENT_ID, 3397
 - HEADER_CONSUMERPRIORITY, 3397
 - HEADER_CONTENTLENGTH, 3397
 - HEADER_CORRELATIONID, 3397
 - HEADER_DESTINATION, 3397
 - HEADER_DISPATCH_ASYNC, 3397
 - HEADER_EXCLUSIVE, 3397
 - HEADER_EXPIRES, 3397
 - HEADER_ID, 3397
 - HEADER_JMSPRIORITY, 3397
 - HEADER_LOGIN, 3397
 - HEADER_MAXPENDINGMSGLIMIT, 3397
 - HEADER_MESSAGE, 3397
 - HEADER_MESSAGEID, 3397
 - HEADER_NOLOCAL, 3397
 - HEADER_OLDSUBSCRIPTIONNAME, 3397
 - HEADER_ORIGINALPASSWORD, 3397
 - HEADER_PERSISTENT, 3397
 - HEADER_PREFETCHSIZE, 3397
 - HEADER_RECEIPT_REQUIRED, 3397
 - HEADER_RECEIPTID, 3397
 - HEADER_REDELIVERED, 3397
 - HEADER_REDELIVERYCOUNT, 3397
 - HEADER_REPLYTO, 3397
 - HEADER_REQUESTID, 3397
 - HEADER_RESPONSEID, 3397
 - HEADER_RETROACTIVE, 3397
 - HEADER_SELECTOR, 3397
 - HEADER_SESSIONID, 3397
 - HEADER_SUBSCRIPTION, 3397
 - HEADER_SUBSCRIPTIONNAME, 3397
 - HEADER_TIMESTAMP, 3397
 - HEADER_TRANSACTIONID, 3397
 - HEADER_TRANSFORMATION, 3397
 - HEADER_TRANSFORMATION_ERROR, 3397
 - HEADER_TYPE, 3397
 - MESSAGE, 3397
 - QUEUE_PREFIX, 3397
 - RECEIPT, 3397
 - SEND, 3397
 - SUBSCRIBE, 3397
 - TEMPQUEUE_PREFIX, 3397
 - TEMPTOPIC_PREFIX, 3397
 - TEXT, 3397
 - TOPIC_PREFIX, 3397
 - UNSUBSCRIBE, 3397
- activemq::wireformat::stomp::StompFrame, 3398
 - ~StompFrame, 3399
 - clone, 3399
 - copy, 3399
 - fromStream, 3400
 - getBody, 3400
 - getBodyLength, 3400
 - getCommand, 3400
 - getProperties, 3400, 3401
 - getProperty, 3401
 - hasProperty, 3401
 - removeProperty, 3401
 - setBody, 3401
 - setCommand, 3402
 - setProperty, 3402

- StompFrame, 3399
- toStream, 3402
- activemq::wireformat::stomp::StompHelper, 3402
 - ~StompHelper, 3404
 - convertConsumerId, 3404
 - convertDestination, 3404
 - convertMessageId, 3405
 - convertProducerId, 3405
 - convertProperties, 3406
 - convertTransactionId, 3406
 - StompHelper, 3404
- activemq::wireformat::stomp::StompWireFormat, 3407
 - ~StompWireFormat, 3408
 - createNegotiator, 3408
 - getVersion, 3408
 - hasNegotiator, 3408
 - inReceive, 3409
 - marshal, 3409
 - setVersion, 3409
 - StompWireFormat, 3408
 - unmarshal, 3409
- activemq::wireformat::stomp::StompWireFormatFactory, 3410
 - ~StompWireFormatFactory, 3411
 - createWireFormat, 3411
 - StompWireFormatFactory, 3411
- activemq::wireformat::WireFormat, 3712
 - ~WireFormat, 3714
 - createNegotiator, 3714
 - getVersion, 3714
 - hasNegotiator, 3714
 - inReceive, 3714
 - marshal, 3715
 - setVersion, 3715
 - unmarshal, 3715
- activemq::wireformat::WireFormatFactory, 3716
 - ~WireFormatFactory, 3717
 - createWireFormat, 3717
- activemq::wireformat::WireFormatNegotiator, 3751
 - ~WireFormatNegotiator, 3752
 - WireFormatNegotiator, 3752
- activemq::wireformat::WireFormatRegistry, 3752
 - ~WireFormatRegistry, 3753
 - findFactory, 3753
 - getInstance, 3753
 - getWireFormatNames, 3754
 - registerFactory, 3754
 - unregisterFactory, 3754
- ActiveMQBlobMessage
 - activemq::commands::ActiveMQBlobMessage, 167
- ActiveMQBlobMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM, 176
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBlobM, 184
 - activemq::wireformat::openwire::marshal::v3::ActiveMQBlobM, 172
 - activemq::wireformat::openwire::marshal::v4::ActiveMQBlobM, 180
 - activemq::wireformat::openwire::marshal::v5::ActiveMQBlobM, 188
 - activemq::wireformat::openwire::marshal::v6::ActiveMQBlobM, 192
- ActiveMQBytesMessage
 - activemq::commands::ActiveMQBytesMessage, 198
- ActiveMQBytesMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesM, 215
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBytesM, 231
 - activemq::wireformat::openwire::marshal::v3::ActiveMQBytesM, 211
 - activemq::wireformat::openwire::marshal::v4::ActiveMQBytesM, 219
 - activemq::wireformat::openwire::marshal::v5::ActiveMQBytesM, 223
 - activemq::wireformat::openwire::marshal::v6::ActiveMQBytesM, 227
- ActiveMQConnection
 - activemq::core::ActiveMQConnection, 238
- ActiveMQConnectionFactory
 - activemq::core::ActiveMQConnectionFactory, 254
- ActiveMQConnectionMetaData
 - activemq::core::ActiveMQConnectionMetaData, 263
- ActiveMQConsumer
 - activemq::core::ActiveMQConsumer, 271
- ActiveMQCPP
 - activemq::library::ActiveMQCPP, 278
- ActiveMQDestination
 - activemq::commands::ActiveMQDestination, 282
- ActiveMQDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestin, 295
 - activemq::wireformat::openwire::marshal::v2::ActiveMQDestin, 307
 - activemq::wireformat::openwire::marshal::v3::ActiveMQDestin, 291

- activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 299
- activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 303
- activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller, 311
- ActiveMQException
 - activemq::exceptions::ActiveMQException, 314
- ActiveMQMapMessage
 - activemq::commands::ActiveMQMapMessage, 319
- ActiveMQMapMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 334
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 346
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 330
 - activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 338
 - activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 342
 - activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller, 350
- ActiveMQMessage
 - activemq::commands::ActiveMQMessage, 353
- ActiveMQMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 360
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 372
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 356
 - activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 364
 - activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 368
 - activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller, 376
- ActiveMQMessageTemplate
 - activemq::commands::ActiveMQMessageTemplate, 383
- ActiveMQObjectMessage
 - activemq::commands::ActiveMQObjectMessage, 397
- ActiveMQObjectMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 404
 - activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 416
 - activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 400
- ActiveMQProducer
 - activemq::core::ActiveMQProducer, 425
- ActiveMQProperties
 - activemq::util::ActiveMQProperties, 432
- ActiveMQQueue
 - activemq::commands::ActiveMQQueue, 436
- ActiveMQQueueBrowser
 - activemq::core::ActiveMQQueueBrowser, 446
- ActiveMQQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 446
 - activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, 458
 - activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 442
 - activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller, 450
- ActiveMQQueueSession
 - activemq::wireformat::openwire::marshal::v5::ActiveMQQueueSession, 454
 - activemq::wireformat::openwire::marshal::v6::ActiveMQQueueSession, 462
- ActiveMQSession
 - activemq::core::ActiveMQSession, 469
- ActiveMQSessionExecutor
 - activemq::core::ActiveMQSessionExecutor, 482
- ActiveMQSessionMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQSessionMarshaller, 483
- ActiveMQStreamMessage
 - activemq::commands::ActiveMQStreamMessage, 505
- ActiveMQStreamMessageMarshaller
 - activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 517
 - activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 501
 - activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 509
 - activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 509
 - activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller, 509
- ActiveMQTempDestination
 - activemq::core::ActiveMQTempDestination, 525

- ActiveMQTempDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 531
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 542
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 528
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 535
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 539
 - activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller, 546
- ActiveMQTempQueue
 - activemq::commands::ActiveMQTempQueue, 550
- ActiveMQTempQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 558
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 570
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 554
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 562
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 566
 - activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller, 574
- ActiveMQTempTopic
 - activemq::commands::ActiveMQTempTopic, 578
- ActiveMQTempTopicMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 590
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 598
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 582
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 586
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 594
 - activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller, 602
- ActiveMQTextMessage
 - activemq::commands::ActiveMQTextMessage, 606
- ActiveMQTextMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 618
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 630
- activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 610
- activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 614
- activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 622
- activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller, 626
- ActiveMQTopic
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 646
- activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 658
- activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 638
- ActiveMQTopicQueueMarshaller
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTopicQueueMarshaller, 642
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTopicQueueMarshaller, 650
 - activemq::wireformat::openwire::marshal::v6::ActiveMQTopicQueueMarshaller, 654
- ActiveMQTransactionContextMarshaller
 - activemq::core::ActiveMQTransactionContext, 661
- add
 - activemq::TempQueueMarshaller::CloseTransportsTask, 1067
- activemq::transport::failover::FailoverTransport, 1755
- decaf::util::AbstractCollection, 146
- decaf::util::AbstractQueue, 159
- decaf::util::Collection, 1099
- decaf::util::List, 2192
- decaf::util::ListIterator, 2198
- decaf::util::PriorityQueue, 2835
- decaf::util::StlList, 3363
- decaf::util::StlSet, 3392
- decaf::util::AbstractQueue, 159
- decaf::util::Collection, 1100
- decaf::util::StlList, 3364
- addAndGet
 - decaf::util::concurrent::atomic::AtomicInteger, 681
- addAsResource
 - ActiveMQTextMessageMarshaller, 2612
- addCommand
 - ActiveMQTextMessageMarshaller, 3624
- addConnection
 - ActiveMQTextMessageMarshaller, 3624

- activemq::cmsutil::ResourceLifecycleManager
 - addTransactionState 3075
- addConsumer
 - activemq::core::ActiveMQSession, 469
 - activemq::state::SessionState, 3219
- addDestination
 - activemq::cmsutil::ResourceLifecycleManager, 3075
- addDispatcher
 - activemq::core::ActiveMQConnection, 238
- addHandler
 - decaf::util::logging::Logger, 2241
- additionalPredicate
 - activemq::commands::ConsumerInfo, 1365
- addLogger
 - decaf::util::logging::LogManager, 2258
- addMarshaller
 - activemq::wireformat::openwire::OpenWireFormat
 - _stream_s, 3795
 - 2703
- addMessageConsumer
 - activemq::cmsutil::ResourceLifecycleManager
 - advisory 3075
- addMessageProducer
 - activemq::cmsutil::ResourceLifecycleManager
 - ADVISORY_PREFIX 3075
- addNetworkResource
 - decaf::internal::net::Network, 2612
- addProducer
 - activemq::core::ActiveMQConnection, 239
 - activemq::core::ActiveMQSession, 469
 - activemq::state::SessionState, 3219
- addProducerState
 - activemq::state::TransactionState, 3624
- addPropertyChangeListener
 - decaf::util::logging::LogManager, 2258
- addResource
 - decaf::internal::util::ResourceLifecycleManager
 - 3073
- address
 - decaf::net::SocketImpl, 3313
- addressBytes
 - decaf::net::InetAddress, 1891
- addSession
 - activemq::cmsutil::ResourceLifecycleManager, 3075
 - activemq::state::ConnectionState, 1292
- addSynchronization
 - activemq::core::ActiveMQTransactionContext
 - allocate 662
- addTask
 - activemq::threads::CompositeTaskRunner, 1134
- addTempDestination
 - activemq::state::ConnectionState, 1292
- activemq::state::ConnectionState, 1292
- addTransportListener
 - activemq::core::ActiveMQConnection, 239
- addURI
 - activemq::transport::CompositeTransport, 1136
 - activemq::transport::failover::FailoverTransport, 1755
 - activemq::transport::failover::URIPool, 3682
- addURIs
 - activemq::transport::failover::URIPool, 3683
- adjustMinimum
 - decaf::internal::util::TimerTaskHeap, 3557
- adler
 - Adler32
 - decaf::util::zip::Adler32, 664
- activemq::commands::ActiveMQDestination, 288
- ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 288
- after
 - decaf::util::Date, 1560
- afterCommit
 - activemq::core::Synchronization, 3477
- afterMarshal
 - activemq::commands::BaseDataStructure, 765
 - activemq::wireformat::MarshalAware, 2329
- afterMessageIsConsumed
 - activemq::core::ActiveMQConsumer, 272
- afterRollback
 - activemq::core::Synchronization, 3477
- afterUnmarshal
 - activemq::commands::BaseDataStructure, 765
 - activemq::commands::Message, 2362
 - activemq::commands::WireFormatInfo, 3720
 - activemq::wireformat::MarshalAware, 2329
- ALL
 - decaf::util::logging::Level, 2189
 - decaf::nio::ByteBuffer, 959
 - decaf::nio::CharBuffer, 1041
 - decaf::nio::DoubleBuffer, 1695
 - decaf::nio::FloatBuffer, 1803
 - decaf::nio::IntBuffer, 1933
 - decaf::nio::LongBuffer, 2294

- decaf::nio::ShortBuffer, 3241
- AMQ_CATCH_ALL_THROW_CMSEXCEPTION
 - CMSExceptionSupport.h, 3887
- AMQ_CATCH_EXCEPTION_CONVERT
 - activemq/exceptions/ExceptionDefines.h, 3855
- AMQ_CATCH_NOTHROW
 - activemq/exceptions/ExceptionDefines.h, 3855
- AMQ_CATCH_RETHROW
 - activemq/exceptions/ExceptionDefines.h, 3855
- AMQ_CATCHALL_NOTHROW
 - activemq/exceptions/ExceptionDefines.h, 3856
- AMQ_CATCHALL_THROW
 - activemq/exceptions/ExceptionDefines.h, 3856
- AMQCPP_API
 - activemq/util/Config.h, 3889
- ANY
 - decaf::net::InetAddress, 1891
- ANY_CHILD
 - activemq::commands::ActiveMQDestination::DestinationFilter, 1613
- ANY_DESCENDENT
 - activemq::commands::ActiveMQDestination::DestinationFilter, 1613
- anyBytes
 - decaf::net::InetAddress, 1891
- append
 - decaf::io::Writer, 3757, 3758
 - decaf::lang::Appendable, 667
 - decaf::nio::CharBuffer, 1041, 1042
- AprPool
 - decaf::internal::AprPool, 669
- array
 - decaf::internal::nio::ByteBuffer, 928
 - decaf::internal::nio::CharArrayBuffer, 1032
 - decaf::internal::nio::DoubleArrayBuffer, 1688
 - decaf::internal::nio::FloatArrayBuffer, 1796
 - decaf::internal::nio::IntArrayBuffer, 1926
 - decaf::internal::nio::LongArrayBuffer, 2286
 - decaf::internal::nio::ShortArrayBuffer, 3234
 - decaf::nio::ByteBuffer, 960
 - decaf::nio::CharBuffer, 1042
 - decaf::nio::DoubleBuffer, 1696
 - decaf::nio::FloatBuffer, 1803
 - decaf::nio::IntBuffer, 1934
 - decaf::nio::LongBuffer, 2294
 - decaf::nio::ShortBuffer, 3241
- arraycopy
 - decaf::lang::System, 3489, 3490
- arrayOffset
 - decaf::internal::nio::ByteBuffer, 928
 - decaf::internal::nio::CharArrayBuffer, 1032
 - decaf::internal::nio::DoubleArrayBuffer, 1688
 - decaf::internal::nio::FloatArrayBuffer, 1796
 - decaf::internal::nio::IntArrayBuffer, 1926
 - decaf::internal::nio::LongArrayBuffer, 2287
 - decaf::internal::nio::ShortArrayBuffer, 3234
 - decaf::nio::ByteBuffer, 960
 - decaf::nio::CharBuffer, 1043
 - decaf::nio::DoubleBuffer, 1696
 - decaf::nio::FloatBuffer, 1803
 - decaf::nio::IntBuffer, 1934
 - decaf::nio::LongBuffer, 2294
 - decaf::nio::ShortBuffer, 3242
- ArrayPointer
 - decaf::lang::ArrayPointer, 672, 673
- arrival
 - activemq::commands::Message, 2374
- asCharBuffer
 - decaf::internal::nio::ByteBuffer, 928
 - decaf::nio::ByteBuffer, 960
- asciiToModifiedUtf8
 - activemq::util::MarshallingSupport, 2336
- asDoubleBuffer
 - decaf::internal::nio::ByteBuffer, 929
 - decaf::nio::ByteBuffer, 961
- asFloatBuffer
 - decaf::internal::nio::ByteBuffer, 929
 - decaf::nio::ByteBuffer, 961
- asIntBuffer
 - decaf::internal::nio::ByteBuffer, 929
 - decaf::nio::ByteBuffer, 961
- asLongBuffer
 - decaf::internal::nio::ByteBuffer, 930
 - decaf::nio::ByteBuffer, 962
- asReadOnlyBuffer
 - decaf::internal::nio::ByteBuffer, 930
 - decaf::internal::nio::CharArrayBuffer, 1033
 - decaf::internal::nio::DoubleArrayBuffer, 1689
 - decaf::internal::nio::FloatArrayBuffer, 1796
 - decaf::internal::nio::IntArrayBuffer, 1927
 - decaf::internal::nio::LongArrayBuffer, 2287
 - decaf::internal::nio::ShortArrayBuffer, 3235
 - decaf::nio::ByteBuffer, 962
 - decaf::nio::CharBuffer, 1043
 - decaf::nio::DoubleBuffer, 1696
 - decaf::nio::FloatBuffer, 1804

- decaf::nio::IntBuffer, 1934
- decaf::nio::LongBuffer, 2295
- decaf::nio::ShortBuffer, 3242
- Assert
 - zutil.h, 4217
- asShortBuffer
 - decaf::internal::nio::ByteBuffer, 931
 - decaf::nio::ByteBuffer, 962
- AsyncSignalReadErrorTask
 - activemq::transport::inactivity::InactivityMonitor, 1877
- AsyncWriteTask
 - activemq::transport::inactivity::InactivityMonitor, 1877
- atEOF
 - decaf::util::zip::InflaterInputStream, 1909
- AtomicBoolean
 - decaf::util::concurrent::atomic::AtomicBoolean, 678
- AtomicInteger
 - decaf::util::concurrent::atomic::AtomicInteger, 681
- AtomicRefCounter
 - decaf::util::concurrent::atomic::AtomicRefCounter, 686
- AtomicReference
 - decaf::util::concurrent::atomic::AtomicReference, 688
- AUTO_ACKNOWLEDGE
 - cms::Session, 3151
- avail_in
 - z_stream_s, 3795
- avail_out
 - z_stream_s, 3795
- available
 - decaf::internal::io::StandardInputStream, 3353
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2678
 - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2697
 - decaf::internal::net::tcp::TcpSocket, 3501
 - decaf::internal::net::tcp::TcpSocketInputStream, 3507
 - decaf::io::BlockingByteArrayInputStream, 773
 - decaf::io::BufferedInputStream, 864
 - decaf::io::ByteArrayInputStream, 947
 - decaf::io::FilterInputStream, 1773
 - decaf::io::InputStream, 1911
 - decaf::io::PushbackInputStream, 2943
 - decaf::net::SocketImpl, 3308
 - decaf::util::zip::InflaterInputStream, 1906
- availableProcessors
 - decaf::lang::System, 3490
- await
 - decaf::util::concurrent::CountDownLatch, 1418, 1419
 - decaf::util::concurrent::locks::Condition, 1158, 1159
 - awaitNanos
 - awaitTermination
 - awaitUntil
 - awaitUninterruptibly
- activemq::transport::failover::BackupTransport, 690
- activemq::transport::failover::BackupTransportPool, 694
- BackupTransportPool
 - activemq::transport::failover::BackupTransportPool, 693
- BAD
 - inflate.h, 4206
- base_dist
 - trees.h, 4208
- base_length
 - trees.h, 4208
- BaseCommand
 - activemq::wireformat::openwire::marshal::v1::BaseCommand, 716
 - activemq::wireformat::openwire::marshal::v2::BaseCommand, 736
 - activemq::wireformat::openwire::marshal::v3::BaseCommand, 702
 - activemq::wireformat::openwire::marshal::v4::BaseCommand, 709
 - activemq::wireformat::openwire::marshal::v5::BaseCommand, 722
 - activemq::wireformat::openwire::marshal::v6::BaseCommand, 729
- before

- decaf::util::Date, 1560
- beforeEnd
 - activemq::core::Synchronization, 3477
- beforeMarshal
 - activemq::commands::ActiveMQMapMessage, 319
 - activemq::commands::ActiveMQTextMessage, 606
 - activemq::commands::BaseDataStructure, 765
 - activemq::commands::Message, 2362
 - activemq::commands::WireFormatInfo, 3720
 - activemq::wireformat::MarshalAware, 2329
- beforeMessageIsConsumed
 - activemq::core::ActiveMQConsumer, 272
- beforeUnmarshal
 - activemq::commands::BaseDataStructure, 765
 - activemq::wireformat::MarshalAware, 2330
- BEGIN
 - activemq::wireformat::stomp::StompCommand, 3397
- begin
 - activemq::core::ActiveMQTransactionContext, 662
- BEST_COMPRESSION
 - decaf::util::zip::Deflater, 1603
- BEST_SPEED
 - decaf::util::zip::Deflater, 1603
- bi_buf
 - internal_state, 1985
- bi_valid
 - internal_state, 1985
- BIG_STRING_TYPE
 - activemq::util::PrimitiveValueNode, 2821
- BINARY_MIME_TYPE
 - activemq::commands::ActiveMQBlobMessage, 170
- bind
 - decaf::internal::net::tcp::TcpSocket, 3501
 - decaf::net::ServerSocket, 3141
 - decaf::net::Socket, 3287
 - decaf::net::SocketImpl, 3308
- BindException
 - decaf::net::BindException, 769, 770
- bitCount
 - decaf::lang::Integer, 1945
 - decaf::lang::Long, 2271
- bits
 - code, 1097
 - inflate_state, 1893
- BL_CODES
 - deflate.h, 4201
- bl_count
 - internal_state, 1985
- bl_desc
 - internal_state, 1985
- bl_tree
 - internal_state, 1985
- block_start
 - internal_state, 1985
- BLOCKED
 - decaf::lang::Thread, 3523
- BlockingByteArrayInputStream
 - decaf::io::BlockingByteArrayInputStream, 772
- Boolean
 - decaf::lang::Boolean, 782
- BOOLEAN_TYPE
 - activemq::util::PrimitiveValueNode, 2821
- BooleanExpression
 - activemq::commands::BooleanExpression, 786
- BooleanStream
 - activemq::wireformat::openwire::utils::BooleanStream, 789
- booleanValue
 - decaf::lang::Boolean, 782
- boolValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2815
- branchQualifier
 - activemq::commands::XATransactionId, 3769
- BrokenBarrierException
 - decaf::util::concurrent::BrokenBarrierException, 791, 792
- BrokerError
 - activemq::commands::BrokerError, 794
- BrokerException
 - activemq::exceptions::BrokerException, 797
- BrokerId
 - activemq::commands::BrokerId, 799
- brokerId
 - activemq::commands::BrokerInfo, 831
- BrokerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 810
 - activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 822
 - activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 802
 - activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 806
 - activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 814

- activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller, 818
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 841
- activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 853
- activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 833
- activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 837
- activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 845
- activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller, 849
- brokerInTime
 - activemq::commands::Message, 2374
- brokerMasterConnector
 - activemq::commands::ConnectionInfo, 1263
- brokerName
 - activemq::commands::BrokerInfo, 831
 - activemq::commands::DiscoveryEvent, 1647
- brokerOutTime
 - activemq::commands::Message, 2374
- brokerPath
 - activemq::commands::ConnectionInfo, 1263
 - activemq::commands::ConsumerInfo, 1365
 - activemq::commands::DestinationInfo, 1618
 - activemq::commands::Message, 2374
 - activemq::commands::ProducerInfo, 2902
- brokerSequenceId
 - activemq::commands::MessageId, 2499
- brokerUploadUrl
 - activemq::commands::BrokerInfo, 831
- brokerURL
 - activemq::commands::BrokerInfo, 831
- Browser
 - activemq::core::ActiveMQQueueBrowser, 441
- browser
 - activemq::commands::ConsumerInfo, 1365
- buf
 - decaf::util::zip::DeflaterOutputStream, 1608
- buff
 - decaf::util::zip::InflaterInputStream, 1909
- Buffer
 - decaf::nio::Buffer, 858
- BufferIdMarshaller, 818
- decaf::io::DataOutputStream, 1476
- BufferedInputStream
 - decaf::io::BufferedInputStream, 863
- BufferedOutputStream
 - decaf::io::BufferedOutputStream, 867
- BufferOverflowException
 - decaf::nio::BufferOverflowException, 880, 881
- BufferUnderflowException
 - decaf::nio::BufferUnderflowException, 883, 884
- buildIncomingCommands
 - activemq::transport::mock::ResponseBuilder, 3080
- buildMessage
 - decaf::lang::Exception, 1715
- buildResponse
 - activemq::transport::mock::ResponseBuilder, 3080
 - activemq::wireformat::openwire::OpenWireResponseBuilder, 2718
- BUSY_STATE
- deflate.h, 4202
- Byte
 - decaf::lang::Byte, 886
 - zconf.h, 4211
- BYTE_ARRAY_TYPE
 - activemq::util::PrimitiveValueNode, 2821
- BYTE_TYPE
 - activemq::util::PrimitiveValueNode, 2821
- ByteArrayAdapter
 - decaf::internal::util::ByteArrayAdapter, 898–900
- ByteArrayBuffer
 - decaf::internal::nio::ByteArrayBuffer, 926, 927
- ByteArrayInputStream
 - decaf::io::ByteArrayInputStream, 946, 947
- ByteArrayOutputStream
 - decaf::io::ByteArrayOutputStream, 952
- byteArrayValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2815
- ByteBuffer
 - decaf::nio::ByteBuffer, 959
- Bytcf
 - zconf.h, 4211
- BYTES
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- bytesToInt

- decaf::net::InetAddress, 1887
- byteValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2815
 - decaf::lang::Byte, 887
 - decaf::lang::Character, 1022
 - decaf::lang::Double, 1675
 - decaf::lang::Float, 1783
 - decaf::lang::Integer, 1945
 - decaf::lang::Long, 2271
 - decaf::lang::Number, 2654
 - decaf::lang::Short, 3223
- CachedConsumer
 - activemq::cmsutil::CachedConsumer, 994
- CachedProducer
 - activemq::cmsutil::CachedProducer, 998
- call
 - decaf::util::concurrent::Callable, 1004
- cancel
 - decaf::util::concurrent::Future, 1841
 - decaf::util::Timer, 3545
 - decaf::util::TimerTask, 3555
- CancellationException
 - decaf::util::concurrent::CancellationException, 1005, 1006
- capacity
 - decaf::nio::Buffer, 858
- cause
 - decaf::lang::Exception, 1718
- ceil
 - decaf::lang::Math, 2342
- CertificateEncodingException
 - decaf::security::cert::CertificateEncodingException, 1010, 1011
- CertificateException
 - decaf::security::cert::CertificateException, 1012, 1013
- CertificateExpiredException
 - decaf::security::cert::CertificateExpiredException, 1014, 1015
- CertificateNotYetValidException
 - decaf::security::cert::CertificateNotYetValidException, 1016
- CertificateParsingException
 - decaf::security::cert::CertificateParsingException, 1018
- CHAR_TYPE
 - activemq::util::PrimitiveValueNode, 2821
- Character
 - decaf::lang::Character, 1021
- CharArrayBuffer
 - decaf::internal::nio::CharArrayBuffer, 1031, 1032
- charAt
 - decaf::lang::CharSequence, 1054
 - decaf::lang::String, 3429
 - decaf::nio::CharBuffer, 1044
- CharBuffer
 - decaf::nio::CharBuffer, 1041
- charf
 - zconf.h, 4211
- charValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2815
- CHECK
 - inflate.h, 4206
- check
 - inflate_state, 1893
- checkClosed
 - decaf::io::InputStreamReader, 1921
 - decaf::io::OutputStreamWriter, 2727
 - decaf::net::ServerSocket, 3142
 - decaf::net::Socket, 3287
- checkConnectionFactory
 - activemq::cmsutil::CmsAccessor, 1069
- checkDestinationResolver
 - activemq::cmsutil::CmsDestinationAccessor, 1073
- CheckedInputStream
 - decaf::util::zip::CheckedInputStream, 1056
- CheckedOutputStream
 - decaf::util::zip::CheckedOutputStream, 1058
- checkMapIsUnmarshalled
 - activemq::commands::ActiveMQMapMessage, 319
- checkResult
 - decaf::internal::net::tcp::TcpSocket, 3501
- checkShutdown
 - activemq::state::ConnectionState, 1292
 - activemq::state::SessionState, 3219
 - activemq::state::TransactionState, 3624
- checkValidity
 - decaf::security::cert::X509Certificate, 3764
- ClassCastException
 - decaf::lang::exceptions::ClassCastException, 1063
- ClassName
 - activemq::commands::BrokerError::StackTraceElement, 3351
- cleanup
 - decaf::internal::AprPool, 669
- clear
 - activemq::core::ActiveMQSessionExecutor, 483
 - activemq::core::MessageDispatchChannel, 2433

- activemq::util::ActiveMQProperties, 432
- activemq::util::PrimitiveValueNode, 2824
- activemq::wireformat::openwire::utils::BooleanSerializer, 789
- cms::CMSProperties, 1080
- decaf::internal::util::ByteArrayAdapter, 901
- decaf::nio::Buffer, 858
- decaf::util::AbstractCollection, 147
- decaf::util::AbstractQueue, 160
- decaf::util::Collection, 1100
- decaf::util::concurrent::ConcurrentStlMap, 1145
- decaf::util::concurrent::SynchronousQueue, 3480
- decaf::util::Map, 2307
- decaf::util::PriorityQueue, 2836
- decaf::util::Properties, 2929
- decaf::util::StlList, 3364
- decaf::util::StlMap, 3374
- decaf::util::StlQueue, 3385
- decaf::util::StlSet, 3393
- clearBody
 - activemq::commands::ActiveMQBytesMessage, 198
 - activemq::commands::ActiveMQMapMessage, 319
 - activemq::commands::ActiveMQMessageTemplate, 383
 - activemq::commands::ActiveMQStreamMessage, 489
 - activemq::commands::ActiveMQTextMessage, 606
 - cms::Message, 2379
- clearMessagesInProgress
 - activemq::core::ActiveMQConsumer, 272
 - activemq::core::ActiveMQSession, 470
 - activemq::core::ActiveMQSessionExecutor, 483
- clearProperties
 - activemq::commands::ActiveMQMessageTemplate, 383
 - cms::Message, 2380
- clearProperty
 - decaf::lang::System, 3491
- CLIENT_ACKNOWLEDGE
 - cms::Session, 3152
- clientId
 - activemq::commands::ConnectionInfo, 1263
 - activemq::commands::JournalTopicAck, 2045
 - activemq::commands::RemoveSubscriptionInfo, 3019
 - activemq::commands::SubscriptionInfo, 3438
 - activemq::commands::ConnectionInfo, 1263
 - clockSequence
 - decaf::util::UUID, 3708
 - clone
 - activemq::commands::ActiveMQBlobMessage, 167
 - activemq::commands::ActiveMQBytesMessage, 198
 - activemq::commands::ActiveMQMapMessage, 319
 - activemq::commands::ActiveMQMessage, 353
 - activemq::commands::ActiveMQObjectMessage, 397
 - activemq::commands::ActiveMQQueue, 436
 - activemq::commands::ActiveMQStreamMessage, 489
 - activemq::commands::ActiveMQTempQueue, 550
 - activemq::commands::ActiveMQTempTopic, 578
 - activemq::commands::ActiveMQTextMessage, 606
 - activemq::commands::ActiveMQTopic, 634
 - activemq::core::policies::DefaultPrefetchPolicy, 1567
 - activemq::core::policies::DefaultRedeliveryPolicy, 1570
 - activemq::core::PrefetchPolicy, 2785
 - activemq::core::RedeliveryPolicy, 2974
 - activemq::exceptions::ActiveMQException, 315
 - activemq::exceptions::BrokerException, 798
 - activemq::util::ActiveMQProperties, 432
 - activemq::wireformat::stomp::StompFrame, 3399
 - cms::BytesMessage, 982
 - cms::CMSProperties, 1080
 - cms::Destination, 1612
 - cms::Message, 2380
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2662
 - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2689
 - decaf::io::EOFException, 1709
 - decaf::io::InterruptedIOException, 1992
 - decaf::io::IOException, 2005

- decaf::io::UnsupportedEncodingException, 3656
- decaf::io::UTFDataFormatException, 3705
- decaf::lang::ArrayPointer, 673
- decaf::lang::Exception, 1715
- decaf::lang::exceptions::ClassCastException, 1064
- decaf::lang::exceptions::IllegalArgumentException, 1865
- decaf::lang::exceptions::IllegalMonitorStateException, 1867
- decaf::lang::exceptions::IllegalStateException, 1871
- decaf::lang::exceptions::IllegalThreadStateException, 1874
- decaf::lang::exceptions::IndexOutOfBoundsException, 1879
- decaf::lang::exceptions::InterruptedException, 1989
- decaf::lang::exceptions::InvalidStateException, 2002
- decaf::lang::exceptions::NoSuchElementException, 2647
- decaf::lang::exceptions::NullPointerException, 2652
- decaf::lang::exceptions::NumberFormatException, 2657
- decaf::lang::exceptions::RuntimeException, 3116
- decaf::lang::exceptions::UnsupportedOperationException, 3659
- decaf::lang::Throwable, 3538
- decaf::net::BindException, 770
- decaf::net::ConnectException, 1167
- decaf::net::HttpRetryException, 1861
- decaf::net::MalformedURLException, 2305
- decaf::net::NoRouteToHostException, 2642
- decaf::net::PortUnreachableException, 2783
- decaf::net::ProtocolException, 2939
- decaf::net::SocketException, 3300
- decaf::net::SocketTimeoutException, 3321
- decaf::net::UnknownHostException, 3651
- decaf::net::UnknownServiceException, 3654
- decaf::net::URISyntaxException, 3689
- decaf::nio::BufferOverflowException, 882
- decaf::nio::BufferUnderflowException, 884
- decaf::nio::InvalidMarkException, 1999
- decaf::nio::ReadOnlyBufferException, 2968
- decaf::security::cert::CertificateEncodingException, 1011
- decaf::security::cert::CertificateException, 1013
- decaf::security::cert::CertificateExpiredException, 1015
- decaf::security::cert::CertificateNotYetValidException, 1017
- decaf::security::cert::CertificateParsingException, 1019
- decaf::security::GeneralSecurityException, 1847
- decaf::security::InvalidKeyException, 1996
- decaf::security::KeyException, 2153
- decaf::security::KeyManagementException, 2155
- decaf::security::NoSuchAlgorithmException, 2645
- decaf::security::NoSuchProviderException, 2650
- decaf::security::SignatureException, 3278
- decaf::util::concurrent::BrokenBarrierException, 792
- decaf::util::concurrent::CancellationException, 1006
- decaf::util::concurrent::ExecutionException, 1749
- decaf::util::concurrent::RejectedExecutionException, 2986
- decaf::util::concurrent::TimeoutException, 3543
- decaf::util::Properties, 2929
- decaf::util::zip::DataFormatException, 3452
- decaf::util::zip::ZipException, 3798
- cloneDataStructure
- activemq::commands::ActiveMQBlobMessage, 168
- activemq::commands::ActiveMQBytesMessage, 198
- activemq::commands::ActiveMQDestination, 282
- activemq::commands::ActiveMQMapMessage, 319
- activemq::commands::ActiveMQMessage, 353
- activemq::commands::ActiveMQObjectMessage, 397
- activemq::commands::ActiveMQQueue, 436
- activemq::commands::ActiveMQStreamMessage, 489
- activemq::commands::ActiveMQTempDestination, 525
- activemq::commands::ActiveMQTempQueue, 550
- activemq::commands::ActiveMQTempTopic, 578

- activemq::commands::ActiveMQTextMessage, 606
- activemq::commands::ActiveMQTopic, 634
- activemq::commands::BooleanExpression, 786
- activemq::commands::BrokerError, 794
- activemq::commands::BrokerId, 799
- activemq::commands::BrokerInfo, 826
- activemq::commands::ConnectionControl, 1173
- activemq::commands::ConnectionError, 1202
- activemq::commands::ConnectionId, 1232
- activemq::commands::ConnectionInfo, 1259
- activemq::commands::ConsumerControl, 1303
- activemq::commands::ConsumerId, 1332
- activemq::commands::ConsumerInfo, 1360
- activemq::commands::ControlCommand, 1391
- activemq::commands::DataArrayResponse, 1424
- activemq::commands::DataResponse, 1478
- activemq::commands::DataStructure, 1554
- activemq::commands::DestinationInfo, 1615
- activemq::commands::DiscoveryEvent, 1645
- activemq::commands::ExceptionResponse, 1721
- activemq::commands::FlushCommand, 1813
- activemq::commands::IntegerResponse, 1957
- activemq::commands::JournalQueueAck, 2015
- activemq::commands::JournalTopicAck, 2042
- activemq::commands::JournalTrace, 2070
- activemq::commands::JournalTransaction, 2096
- activemq::commands::KeepAliveInfo, 2123
- activemq::commands::LastPartialCommand, 2157
- activemq::commands::LocalTransactionId, 2202
- activemq::commands::Message, 2362
- activemq::commands::MessageAck, 2395
- activemq::commands::MessageDispatch, 2428
- activemq::commands::MessageDispatchNotification, 2463
- activemq::commands::MessageId, 2496
- activemq::commands::MessagePull, 2565
- activemq::commands::NetworkBridgeFilter, 2615
- activemq::commands::PartialCommand, 2729
- activemq::commands::ProducerAck, 2840
- activemq::commands::ProducerId, 2871
- activemq::commands::ProducerInfo, 2899
- activemq::commands::RemoveInfo, 2989
- activemq::commands::RemoveSubscriptionInfo, 3016
- activemq::commands::ReplayCommand, 3044
- activemq::commands::Response, 3077
- activemq::commands::SessionId, 3163
- activemq::commands::SessionInfo, 3189
- activemq::commands::ShutdownInfo, 3250
- activemq::commands::SubscriptionInfo, 3435
- activemq::commands::TransactionId, 3571
- activemq::commands::TransactionInfo, 3596
- activemq::commands::WireFormatInfo, 3720
- activemq::commands::XATransactionId, 3766
- close
 - activemq::cmsutil::CachedConsumer, 994
 - activemq::cmsutil::CachedProducer, 998
 - activemq::cmsutil::PooledSession, 2768
 - activemq::commands::ActiveMQTempDestination, 525
 - activemq::commands::ConnectionControl, 1176
 - activemq::commands::ConsumerControl, 1306
 - activemq::core::ActiveMQConnection, 239
 - activemq::core::ActiveMQConsumer, 272
 - activemq::core::ActiveMQProducer, 425
 - activemq::core::ActiveMQQueueBrowser, 440
 - activemq::core::ActiveMQSession, 470
 - activemq::core::ActiveMQSessionExecutor, 483
 - activemq::core::MessageDispatchChannel, 2433
 - activemq::transport::correlator::ResponseCorrelator, 3082
 - activemq::transport::failover::FailoverTransport, 1755
 - activemq::transport::inactivity::InactivityMonitor, 1875
 - activemq::transport::IOTransport, 2007

- activemq::transport::mock::MockTransport, 2594
- activemq::transport::tcp::TcpTransport, 3512
- activemq::transport::TransportFilter, 3638
- activemq::wireformat::openwire::OpenWireFormat, 2714
- cms::Closeable, 1065
- cms::Connection, 1169
- cms::Session, 3152
- decaf::internal::io::StandardErrorOutputStream, 3352
- decaf::internal::io::StandardOutputStream, 3354
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2679
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2697
- decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2699
- decaf::internal::net::tcp::TcpSocket, 3501
- decaf::internal::net::tcp::TcpSocketInputStream, 3507
- decaf::internal::net::tcp::TcpSocketOutputStream, 3510
- decaf::io::BlockingByteArrayInputStream, 773
- decaf::io::BufferedInputStream, 864
- decaf::io::Closeable, 1066
- decaf::io::FilterInputStream, 1774
- decaf::io::FilterOutputStream, 1778
- decaf::io::InputStream, 1912
- decaf::io::InputStreamReader, 1921
- decaf::io::OutputStream, 2720
- decaf::io::OutputStreamWriter, 2727
- decaf::net::ServerSocket, 3142
- decaf::net::Socket, 3287
- decaf::net::SocketImpl, 3309
- decaf::util::logging::ConsoleHandler, 1301
- decaf::util::logging::StreamHandler, 3414
- decaf::util::zip::DeflaterOutputStream, 1607
- decaf::util::zip::InflaterInputStream, 1906
- CLOSE_FAILURE
 - decaf::util::logging::ErrorManager, 1711
- closed
 - decaf::io::FilterInputStream, 1777
 - decaf::io::FilterOutputStream, 1780
- CloseTransportsTask
 - activemq::transport::failover::CloseTransportsTask, 1067
- cluster
 - activemq::commands::Message, 2374
- cms, 117
 - cms/Config.h, CMS_API, 3889
 - cms::BytesMessage, 979
 - ~BytesMessage, 982
 - clone, 982
 - getBodyBytes, 982
 - getBodyLength, 983
 - readBoolean, 983
 - readByte, 983
 - readBytes, 984
 - readChar, 985
 - readDouble, 985
 - readFloat, 986
 - readInt, 986
 - readLong, 986
 - readShort, 987
 - readString, 987
 - readUnsignedShort, 987
 - readUTF, 988
 - reset, 988
 - setBodyBytes, 988
 - writeBoolean, 988
 - writeByte, 989
 - writeBytes, 989
 - writeChar, 990
 - writeDouble, 990
 - writeFloat, 990
 - writeInt, 991
 - writeLong, 991
 - writeShort, 991
 - writeString, 992
 - writeUnsignedShort, 992
 - writeUTF, 992
 - cms::Closeable, 1064
 - ~Closeable, 1065
 - close, 1065
 - cms::CMSException, 1074
 - ~CMSException, 1076
 - CMSException, 1076
 - getCause, 1076
 - getMessage, 1076
 - getStackTrace, 1076
 - getStackTraceString, 1076
 - printStackTrace, 1077
 - setMark, 1077
 - what, 1077
 - cms::CMSProperties, 1079
 - ~CMSProperties, 1080
 - clear, 1080
 - clone, 1080
 - copy, 1080
 - getProperty, 1080
 - hasProperty, 1081
 - isEmpty, 1081

- remove, 1081
- setProperty, 1081
- toArray, 1082
- toString, 1082
- cms::CMSSecurityException, 1082
 - ~CMSSecurityException, 1083
 - CMSSecurityException, 1083
- cms::Connection, 1168
 - ~Connection, 1169
 - close, 1169
 - createSession, 1170
 - getClientID, 1170
 - getExceptionListener, 1170
 - getMetaData, 1170
 - setClientID, 1171
 - setExceptionListener, 1171
- cms::ConnectionFactory, 1228
 - ~ConnectionFactory, 1229
 - createCMSConnectionFactory, 1229
 - createConnection, 1229, 1230
- cms::ConnectionMetaData, 1287
 - ~ConnectionMetaData, 1288
 - getCMSMajorVersion, 1288
 - getCMSMinorVersion, 1289
 - getCMSProviderName, 1289
 - getCMSVersion, 1289
 - getCMSXPropertyNames, 1289
 - getProviderMajorVersion, 1290
 - getProviderMinorVersion, 1290
 - getProviderVersion, 1290
- cms::DeliveryMode, 1609
 - ~DeliveryMode, 1610
 - DELIVERY_MODE, 1610
 - NON_PERSISTENT, 1610
 - PERSISTENT, 1610
- cms::Destination, 1610
 - ~Destination, 1612
 - clone, 1612
 - copy, 1612
 - DestinationType, 1611
 - getCMSProperties, 1612
 - getDestinationType, 1612
 - QUEUE, 1611
 - TEMPORARY_QUEUE, 1611
 - TEMPORARY_TOPIC, 1611
 - TOPIC, 1611
- cms::ExceptionListener, 1719
 - ~ExceptionListener, 1719
 - onException, 1719
- cms::IllegalStateException, 1868
 - ~IllegalStateException, 1869
 - IllegalStateException, 1869
- cms::InvalidClientIdException, 1992
 - ~InvalidClientIdException, 1993
 - InvalidClientIdException, 1993
- cms::InvalidDestinationException, 1993
 - ~InvalidDestinationException, 1994
 - InvalidDestinationException, 1994
- cms::InvalidSelectorException, 1999
 - ~InvalidSelectorException, 2000
 - InvalidSelectorException, 2000
- cms::MapMessage, 2318
 - ~MapMessage, 2320
 - getBoolean, 2320
 - getByte, 2321
 - getBytes, 2321
 - getChar, 2321
 - getDouble, 2321
 - getFloat, 2322
 - getInt, 2322
 - getLong, 2322
 - getMapNames, 2323
 - getShort, 2323
 - getString, 2323
 - itemExists, 2323
 - setBoolean, 2324
 - setByte, 2324
 - setBytes, 2324
 - setChar, 2325
 - setDouble, 2325
 - setFloat, 2325
 - setInt, 2326
 - setLong, 2326
 - setShort, 2326
 - setString, 2327
- cms::Message, 2375
 - ~Message, 2379
 - acknowledge, 2379
 - clearBody, 2379
 - clearProperties, 2380
 - clone, 2380
 - getBooleanProperty, 2380
 - getByteProperty, 2380
 - getCMSCorrelationID, 2381
 - getCMSDeliveryMode, 2381
 - getCMSDestination, 2381
 - getCMSExpiration, 2382
 - getCMSMessageID, 2382
 - getCMSPriority, 2383
 - getCMSRedelivered, 2383
 - getCMSReplyTo, 2383
 - getCMSTimestamp, 2384
 - getCMSType, 2384
 - getDoubleProperty, 2384
 - getFloatProperty, 2385
 - getIntProperty, 2385
 - getLongProperty, 2385
 - getPropertyNames, 2386

- getShortProperty, 2386
- getStringProperty, 2386
- propertyExists, 2387
- setBooleanProperty, 2387
- setByteProperty, 2387
- setCMSCorrelationID, 2388
- setCMSDeliveryMode, 2388
- setCMSDestination, 2389
- setCMSExpiration, 2389
- setCMSMessageID, 2389
- setCMSPriority, 2390
- setCMSRedelivered, 2390
- setCMSReplyTo, 2390
- setCMSTimestamp, 2391
- setCMSType, 2391
- setDoubleProperty, 2392
- setFloatProperty, 2392
- setIntProperty, 2392
- setLongProperty, 2392
- setShortProperty, 2393
- setStringProperty, 2393
- cms::MessageConsumer, 2423
 - ~MessageConsumer, 2424
 - getMessageListener, 2424
 - getMessageSelector, 2424
 - receive, 2424, 2425
 - receiveNoWait, 2425
 - setMessageListener, 2425
- cms::MessageEnumeration, 2490
 - ~MessageEnumeration, 2491
 - hasMoreMessages, 2491
 - nextMessage, 2491
- cms::MessageEOFException, 2492
 - ~MessageEOFException, 2493
 - MessageEOFException, 2493
- cms::MessageFormatException, 2493
 - ~MessageFormatException, 2494
 - MessageFormatException, 2494
- cms::MessageListener, 2522
 - ~MessageListener, 2523
 - onMessage, 2523
- cms::MessageNotReadableException, 2548
 - ~MessageNotReadableException, 2549
 - MessageNotReadableException, 2549
- cms::MessageNotWritableException, 2549
 - ~MessageNotWritableException, 2550
 - MessageNotWritableException, 2550
- cms::MessageProducer, 2550
 - ~MessageProducer, 2552
 - getDeliveryMode, 2552
 - getDisableMessageID, 2552
 - getDisableMessageTimeStamp, 2552
 - getPriority, 2553
 - getTimeToLive, 2553
 - send, 2553–2555
 - setDeliveryMode, 2555
 - setDisableMessageID, 2556
 - setDisableMessageTimeStamp, 2556
 - setPriority, 2556
 - setTimeToLive, 2557
- cms::ObjectMessage, 2658
 - ~ObjectMessage, 2658
- cms::Queue, 2947
 - ~Queue, 2947
 - getQueueName, 2947
- cms::QueueBrowser, 2951
 - ~QueueBrowser, 2952
 - getEnumeration, 2952
 - getMessageSelector, 2952
 - getQueue, 2952
- cms::Session, 3148
 - ~Session, 3152
 - AcknowledgeMode, 3151
 - AUTO_ACKNOWLEDGE, 3151
 - CLIENT_ACKNOWLEDGE, 3152
 - close, 3152
 - commit, 3152
 - createBrowser, 3152, 3153
 - createBytesMessage, 3153
 - createConsumer, 3154
 - createDurableConsumer, 3155
 - createMapMessage, 3156
 - createMessage, 3156
 - createProducer, 3156
 - createQueue, 3156
 - createStreamMessage, 3157
 - createTemporaryQueue, 3157
 - createTemporaryTopic, 3157
 - createTextMessage, 3157, 3158
 - createTopic, 3158
 - DUPS_OK_ACKNOWLEDGE, 3152
 - getAcknowledgeMode, 3158
 - INDIVIDUAL_ACKNOWLEDGE, 3152
 - isTransacted, 3159
 - recover, 3159
 - rollback, 3159
 - SESSION_TRANSACTED, 3152
 - unsubscribe, 3160
- cms::Startable, 3355
 - ~Startable, 3356
 - start, 3356
- cms::Stoppable, 3411
 - ~Stoppable, 3412
 - stop, 3412
- cms::StreamMessage, 3415
 - ~StreamMessage, 3418
 - readBoolean, 3418
 - readByte, 3418

- readBytes, 3419
- readChar, 3420
- readDouble, 3420
- readFloat, 3421
- readInt, 3421
- readLong, 3422
- readShort, 3422
- readString, 3422
- readUnsignedShort, 3423
- writeBoolean, 3423
- writeByte, 3423
- writeBytes, 3424
- writeChar, 3424
- writeDouble, 3425
- writeFloat, 3425
- writeInt, 3425
- writeLong, 3426
- writeShort, 3426
- writeString, 3426
- writeUnsignedShort, 3427
- cms::TemporaryQueue, 3516
 - ~TemporaryQueue, 3517
 - destroy, 3517
 - getQueueName, 3517
- cms::TemporaryTopic, 3517
 - ~TemporaryTopic, 3518
 - destroy, 3518
 - getTopicName, 3518
- cms::TextMessage, 3519
 - ~TextMessage, 3519
 - getText, 3519
 - setText, 3520
- cms::Topic, 3568
 - ~Topic, 3568
 - getTopicName, 3568
- cms::UnsupportedOperationException, 3659
 - ~UnsupportedOperationException, 3660
 - UnsupportedOperationException, 3660
- CMS_API
 - cms/Config.h, 3889
- CmsAccessor
 - activemq::cmsutil::CmsAccessor, 1069
- CmsDestinationAccessor
 - activemq::cmsutil::CmsDestinationAccessor, 1073
- CMSException
 - cms::CMSException, 1076
- CMSExceptionSupport.h
 - AMQ_CATCH_ALL_THROW_-CMSEXCEPTION, 3887
- CMSSecurityException
 - cms::CMSSecurityException, 1083
- CmsTemplate
 - activemq::cmsutil::CmsTemplate, 1087
- Code
 - deflate.h, 4202
- code, 1096
 - bits, 1097
 - ct_data_s, 1423
 - op, 1097
 - val, 1097
- CODELENS
 - inflate.h, 4206
- CODES
 - inftrees.h, 4207
- codes
 - inflate_state, 1893
- codetype
 - inftrees.h, 4207
- comm_max
 - gz_header_s, 1849
- command
 - activemq::commands::ControlCommand, 1393
- commandId
 - activemq::commands::PartialCommand, 2731
- COMMENT
 - inflate.h, 4206
- comment
 - gz_header_s, 1849
- COMMENT_STATE
 - deflate.h, 4202
- COMMIT
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- commit
 - activemq::cmsutil::PooledSession, 2768
 - activemq::core::ActiveMQConsumer, 272
 - activemq::core::ActiveMQSession, 470
 - activemq::core::ActiveMQTransactionContext, 662
 - cms::Session, 3152
- compact
 - decaf::internal::nio::ByteBuffer, 931
 - decaf::internal::nio::CharArrayBuffer, 1033
 - decaf::internal::nio::DoubleArrayBuffer, 1689
 - decaf::internal::nio::FloatArrayBuffer, 1797
 - decaf::internal::nio::IntArrayBuffer, 1927
 - decaf::internal::nio::LongArrayBuffer, 2287
 - decaf::internal::nio::ShortArrayBuffer, 3235
 - decaf::nio::ByteBuffer, 963
 - decaf::nio::CharBuffer, 1044
 - decaf::nio::DoubleBuffer, 1697
 - decaf::nio::FloatBuffer, 1804
 - decaf::nio::IntBuffer, 1935

- decaf::nio::LongBuffer, 2295
- decaf::nio::ShortBuffer, 3243
- COMPARATOR
 - activemq::commands::BrokerId, 799
 - activemq::commands::ConnectionId, 1232
 - activemq::commands::ConsumerId, 1332
 - activemq::commands::LocalTransactionId, 2202
 - activemq::commands::MessageId, 2496
 - activemq::commands::ProducerId, 2871
 - activemq::commands::SessionId, 3163
 - activemq::commands::TransactionId, 3571
 - activemq::commands::XATransactionId, 3766
- comparator
 - decaf::util::PriorityQueue, 2836
- compare
 - activemq::util::IdGenerator, 1862
 - decaf::lang::ArrayPointerComparator, 677
 - decaf::lang::Double, 1675
 - decaf::lang::Float, 1783
 - decaf::lang::PointerComparator, 2764
 - decaf::util::Comparator, 1128
 - decaf::util::comparators::Less, 2183
- compareAndSet
 - decaf::util::concurrent::atomic::AtomicBoolean, 678
 - decaf::util::concurrent::atomic::AtomicInteger, 682
 - decaf::util::concurrent::atomic::AtomicReference, 688
- compareTo
 - activemq::commands::BrokerId, 799
 - activemq::commands::ConnectionId, 1232
 - activemq::commands::ConsumerId, 1332
 - activemq::commands::LocalTransactionId, 2202
 - activemq::commands::MessageId, 2496
 - activemq::commands::ProducerId, 2871
 - activemq::commands::SessionId, 3163
 - activemq::commands::TransactionId, 3571
 - activemq::commands::XATransactionId, 3766
 - decaf::lang::Boolean, 782
 - decaf::lang::Byte, 887
 - decaf::lang::Character, 1022
 - decaf::lang::Comparable, 1126
 - decaf::lang::Double, 1675
 - decaf::lang::Float, 1783, 1784
 - decaf::lang::Integer, 1945
 - decaf::lang::Long, 2271
 - decaf::lang::Short, 3223
 - decaf::net::URI, 3665
 - decaf::nio::ByteBuffer, 963
 - decaf::nio::CharBuffer, 1044
 - decaf::nio::DoubleBuffer, 1697
 - decaf::nio::FloatBuffer, 1805
 - decaf::nio::IntBuffer, 1935
 - decaf::nio::LongBuffer, 2295
 - decaf::nio::ShortBuffer, 3243
 - decaf::util::concurrent::TimeUnit, 3561
 - decaf::util::Date, 1561
 - decaf::util::logging::Level, 2188
 - decaf::util::UUID, 3708
- COMPOSITE_SEPARATOR
 - activemq::commands::ActiveMQDestination, 288
- CompositeData
 - activemq::util::CompositeData, 1131
- CompositeTaskRunner
 - activemq::threads::CompositeTaskRunner, 1134
- compressed
 - activemq::commands::Message, 2374
- Concurrent.h
 - synchronized, 4287
 - WAIT_INFINITE, 4287
- ConcurrentStlMap
 - decaf::util::concurrent::ConcurrentStlMap, 1145
- condition
 - decaf::util::concurrent::ConditionHandle, 1163
 - ConditionHandle
 - decaf::util::concurrent::ConditionHandle, 1163
- CONFIG
 - decaf::util::logging::Level, 2189
- config
 - decaf::util::logging::Logger, 2241
- configure
 - activemq::core::PrefetchPolicy, 2785
 - activemq::core::RedeliveryPolicy, 2974
 - activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 2334
 - activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2335
 - activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 2332
 - activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 2333
 - activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2333
 - activemq::wireformat::openwire::marshal::v6::MarshallerFactory, 2331
- configureSocket
 - activemq::transport::tcp::SslTransport, 3349

- activemq::transport::tcp::TcpTransport, 3512
- CONNECT
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- connect
 - activemq::transport::tcp::TcpTransport, 3512
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2679
 - decaf::internal::net::tcp::TcpSocket, 3501
 - decaf::net::Socket, 3287, 3288
 - decaf::net::SocketImpl, 3309
- CONNECTED
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- connectedBrokers
 - activemq::commands::ConnectionControl, 1176
- ConnectException
 - decaf::net::ConnectException, 1166, 1167
- connection
 - activemq::commands::ActiveMQTempDestination, 526
 - activemq::commands::Message, 2374
- CONNECTION_ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 288
- CONNECTION_ALWAYS_SYNC_SEND
 - activemq::core::ActiveMQConstants, 268
- CONNECTION_CLOSE_TIMEOUT
 - activemq::core::ActiveMQConstants, 268
- CONNECTION_DISPATCH_ASYNC
 - activemq::core::ActiveMQConstants, 268
- CONNECTION_PRODUCER_WINDOW_SIZE
 - activemq::core::ActiveMQConstants, 268
- CONNECTION_SEND_TIMEOUT
 - activemq::core::ActiveMQConstants, 268
- CONNECTION_USE_ASYNC_SEND
 - activemq::core::ActiveMQConstants, 268
- CONNECTION_USE_COMPRESSION
 - activemq::core::ActiveMQConstants, 268
- ConnectionControl
 - activemq::commands::ConnectionControl, 1173
- ConnectionControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1186
 - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1198
 - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1178
 - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1182
- activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1190
- activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 1194
- ConnectionError
 - activemq::commands::ConnectionError, 1202
- ConnectionErrorMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1217
 - activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1205
 - activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1209
 - activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1213
 - activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1221
 - activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller, 1225
- ConnectionId
 - activemq::commands::ConnectionId, 1232
- connectionId
 - activemq::commands::BrokerInfo, 831
 - activemq::commands::ConnectionError, 1204
 - activemq::commands::ConnectionInfo, 1263
 - activemq::commands::ConsumerId, 1334
 - activemq::commands::DestinationInfo, 1618
 - activemq::commands::LocalTransactionId, 2204
 - activemq::commands::ProducerId, 2873
 - activemq::commands::RemoveSubscriptionInfo, 3019
 - activemq::commands::SessionId, 3165
 - activemq::commands::TransactionInfo, 3598
- ConnectionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1247
 - activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1235
 - activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1249
 - activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1248
 - activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1251
 - activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller, 1255
- ConnectionInfo

- activemq::commands::ConnectionInfo, 1259
- ConnectionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1277
 - activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1265
 - activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1269
 - activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1273
 - activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1281
 - activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller, 1285
- connectionInterruptProcessingComplete
 - activemq::state::ConnectionStateTracker, 1296
- ConnectionState
 - activemq::state::ConnectionState, 1292
- ConnectionStateTracker
 - activemq::state::ConnectionStateTracker, 1296
- ConsoleHandler
 - decaf::util::logging::ConsoleHandler, 1301
- const
 - zconf.h, 4211
- ConstReferenceType
 - decaf::lang::ArrayPointer, 672
- CONSUMER_ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 288
- CONSUMER_DISPATCHASYNC
 - activemq::core::ActiveMQConstants, 267
- CONSUMER_EXCLUSIVE
 - activemq::core::ActiveMQConstants, 267
- CONSUMER_NOLOCAL
 - activemq::core::ActiveMQConstants, 267
- CONSUMER_PREFETCHSIZE
 - activemq::core::ActiveMQConstants, 267
- CONSUMER_PRIORITY
 - activemq::core::ActiveMQConstants, 267
- CONSUMER_RETROACTIVE
 - activemq::core::ActiveMQConstants, 267
- CONSUMER_SELECTOR
 - activemq::core::ActiveMQConstants, 267
- ConsumerControl
 - activemq::commands::ConsumerControl, 1303
- ConsumerControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1320
 - activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1308
- activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1312
- activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1316
- activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1324
- activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller, 1328
- ConsumerId
 - activemq::commands::ConsumerId, 1332
- consumerIdMarshaller
 - activemq::commands::ConsumerControl, 1306
 - activemq::commands::ConsumerInfo, 1365
- activemq::commands::MessageAck, 2399
- activemq::commands::MessageDispatch, 2431
- activemq::commands::MessageDispatchNotification, 2466
- activemq::commands::MessagePull, 2568
- ConsumerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1347
 - activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1335
 - activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1339
 - activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1343
 - activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1351
 - activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 1355
- ConsumerInfo
 - activemq::commands::ConsumerInfo, 1360
- ConsumerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1379
 - activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1367
 - activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1371
 - activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1375
 - activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1383
 - activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1387
- ConsumerState
 - activemq::state::ConsumerState, 1390
- contains
 - decaf::util::Collection, 148
- decaf::util::Collection, 1101

- decaf::util::concurrent::SynchronousQueue, 3480
- decaf::util::StlList, 3365
- decaf::util::StlSet, 3393
- containsAll
 - decaf::util::AbstractCollection, 148
 - decaf::util::Collection, 1102
 - decaf::util::concurrent::SynchronousQueue, 3480
- containsKey
 - decaf::util::concurrent::ConcurrentStlMap, 1145
 - decaf::util::Map, 2308
 - decaf::util::StlMap, 3374
- containsValue
 - decaf::util::concurrent::ConcurrentStlMap, 1146
 - decaf::util::Map, 2308
 - decaf::util::StlMap, 3374
- content
 - activemq::commands::Message, 2374
- ControlCommand
 - activemq::commands::ControlCommand, 1391
- ControlCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1406
 - activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1394
 - activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1398
 - activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1402
 - activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1410
 - activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller, 1414
- convert
 - activemq::util::PrimitiveValueConverter, 2817
 - decaf::util::concurrent::TimeUnit, 3562
- convertConsumerId
 - activemq::wireformat::stomp::StompHelper, 3404
- convertDestination
 - activemq::wireformat::stomp::StompHelper, 3404
- convertMessageId
 - activemq::wireformat::stomp::StompHelper, 3405
- convertProducerId
 - activemq::wireformat::stomp::StompHelper, 3405
- convertProperties
 - activemq::wireformat::stomp::StompHelper, 3406
- convertToCMSException
 - activemq::exceptions::ActiveMQException, 315
- convertTransactionId
 - activemq::wireformat::stomp::StompHelper, 3406
- COPY
 - gzguts.h, 4204
 - inflate.h, 4206
- copy
 - activemq::commands::ActiveMQQueue, 436
 - activemq::commands::ActiveMQTempQueue, 550
 - activemq::commands::ActiveMQTempTopic, 578
 - activemq::commands::ActiveMQTopic, 634
 - activemq::util::ActiveMQProperties, 432
 - activemq::wireformat::stomp::StompFrame, 3399
 - cms::CMSProperties, 1080
 - cms::Destination, 1612
 - decaf::util::AbstractCollection, 149
 - decaf::util::concurrent::ConcurrentStlMap, 1146
 - decaf::util::Map, 2309
 - decaf::util::Properties, 2929
 - decaf::util::StlList, 3365
 - decaf::util::StlMap, 3375
 - decaf::util::StlSet, 3394
- COPY-CommandMarshaller, inflate.h, 4206
- copyDataStructure
 - activemq::commands::ActiveMQBlobMessage, 168
 - activemq::commands::ActiveMQBytesMessage, 198
 - activemq::commands::ActiveMQDestination, 282
 - activemq::commands::ActiveMQMapMessage, 320
 - activemq::commands::ActiveMQMessage, 354
 - activemq::commands::ActiveMQObjectMessage, 398
 - activemq::commands::ActiveMQQueue, 437
 - activemq::commands::ActiveMQStreamMessage, 489
 - activemq::commands::ActiveMQTempDestination, 525

- activemq::commands::ActiveMQTempQueue, 550
- activemq::commands::ActiveMQTempTopic, 578
- activemq::commands::ActiveMQTextMessage, 607
- activemq::commands::ActiveMQTopic, 635
- activemq::commands::BaseCommand, 696
- activemq::commands::BaseDataStructure, 766
- activemq::commands::BooleanExpression, 786
- activemq::commands::BrokerError, 794
- activemq::commands::BrokerId, 799
- activemq::commands::BrokerInfo, 826
- activemq::commands::ConnectionControl, 1173
- activemq::commands::ConnectionError, 1202
- activemq::commands::ConnectionId, 1232
- activemq::commands::ConnectionInfo, 1259
- activemq::commands::ConsumerControl, 1303
- activemq::commands::ConsumerId, 1332
- activemq::commands::ConsumerInfo, 1360
- activemq::commands::ControlCommand, 1391
- activemq::commands::DataArrayResponse, 1424
- activemq::commands::DataResponse, 1478
- activemq::commands::DataStructure, 1555
- activemq::commands::DestinationInfo, 1615
- activemq::commands::DiscoveryEvent, 1645
- activemq::commands::ExceptionResponse, 1721
- activemq::commands::FlushCommand, 1813
- activemq::commands::IntegerResponse, 1957
- activemq::commands::JournalQueueAck, 2015
- activemq::commands::JournalTopicAck, 2042
- activemq::commands::JournalTrace, 2070
- activemq::commands::JournalTransaction, 2097
- activemq::commands::KeepAliveInfo, 2123
- activemq::commands::LastPartialCommand, 2157
- activemq::commands::LocalTransactionId, 2202
- activemq::commands::Message, 2363
- activemq::commands::MessageAck, 2395
- activemq::commands::MessageDispatch, 2428
- activemq::commands::MessageDispatchNotification, 2463
- activemq::commands::MessageId, 2496
- activemq::commands::MessagePull, 2565
- activemq::commands::NetworkBridgeFilter, 2615
- activemq::commands::PartialCommand, 2729
- activemq::commands::ProducerAck, 2840
- activemq::commands::ProducerId, 2871
- activemq::commands::ProducerInfo, 2899
- activemq::commands::RemoveInfo, 2989
- activemq::commands::RemoveSubscriptionInfo, 3016
- activemq::commands::ReplayCommand, 3044
- activemq::commands::Response, 3077
- activemq::commands::SessionId, 3163
- activemq::commands::SessionInfo, 3190
- activemq::commands::ShutdownInfo, 3250
- activemq::commands::SubscriptionInfo, 3435
- activemq::commands::TransactionId, 3571
- activemq::commands::TransactionInfo, 3596
- activemq::commands::WireFormatInfo, 3721
- activemq::commands::XATransactionId, 3766
- correlationId
 - activemq::commands::Message, 2374
 - activemq::commands::MessagePull, 2568
 - activemq::commands::Response, 3079
- countDown
 - decaf::util::concurrent::CountDownLatch, 1419
- CountDownLatch
 - decaf::util::concurrent::CountDownLatch, 1417
- CounterType
 - decaf::lang::ArrayPointer, 672
 - decaf::lang::Pointer, 2758
- countTokens
 - decaf::util::StringTokenizer, 3431
- CRC32
 - decaf::util::zip::CRC32, 1421
- crc32.h
 - crc_table, 4199
- crc_table
 - crc32.h, 4199

- create
 - activemq::transport::failover::FailoverTransportFactory, 1764
 - activemq::transport::mock::MockTransportFactory, 2603
 - activemq::transport::tcp::TcpTransportFactory, 3515
 - activemq::transport::TransportFactory, 3635
 - activemq::util::CMSExceptionSupport, 1078
 - decaf::internal::net::tcp::TcpSocket, 3502
 - decaf::internal::util::concurrent::ConditionImpl, 1164
 - decaf::internal::util::concurrent::MutexImpl, 2610
 - decaf::net::SocketImpl, 3309
 - decaf::net::URI, 3665
- createBrowser
 - activemq::cmsutil::PooledSession, 2768
 - activemq::core::ActiveMQSession, 470, 471
 - cms::Session, 3152, 3153
- createByteBuffer
 - decaf::internal::nio::BufferFactory, 871, 872
- createBytesMessage
 - activemq::cmsutil::PooledSession, 2769
 - activemq::core::ActiveMQSession, 471
 - cms::Session, 3153
- createCachedConsumer
 - activemq::cmsutil::PooledSession, 2769
- createCachedProducer
 - activemq::cmsutil::PooledSession, 2770
- createCharBuffer
 - decaf::internal::nio::BufferFactory, 872, 873
- createCMSConnectionFactory
 - cms::ConnectionFactory, 1229
- createComposite
 - activemq::transport::failover::FailoverTransportFactory, 1765
 - activemq::transport::mock::MockTransportFactory, 2603
 - activemq::transport::tcp::TcpTransportFactory, 3515
 - activemq::transport::TransportFactory, 3635
- createConnection
 - activemq::cmsutil::CmsAccessor, 1069
 - activemq::core::ActiveMQConnectionFactory, 254, 255
 - cms::ConnectionFactory, 1229, 1230
- createConsumer
 - activemq::cmsutil::PooledSession, 2770, 2771
 - activemq::core::ActiveMQSession, 472
- cms::Session, 3154
- createDestination
 - activemq::commands::ActiveMQDestination, 282
- createDoubleBuffer
 - decaf::internal::nio::BufferFactory, 873, 874
- createDurableConsumer
 - activemq::cmsutil::PooledSession, 2771
 - activemq::core::ActiveMQSession, 473
 - cms::Session, 3155
- createFloatBuffer
 - decaf::internal::nio::BufferFactory, 875
- createIntBuffer
 - decaf::internal::nio::BufferFactory, 876, 877
- createLongBuffer
 - decaf::internal::nio::BufferFactory, 877, 878
- createMapMessage
 - activemq::cmsutil::PooledSession, 2772
 - activemq::core::ActiveMQSession, 473
 - cms::Session, 3156
- createMessage
 - activemq::cmsutil::MessageCreator, 2426
 - activemq::cmsutil::PooledSession, 2772
 - activemq::core::ActiveMQSession, 473
 - cms::Session, 3156
- createMessageEOFException
 - activemq::util::CMSExceptionSupport, 1078
- createMessageFormatException
 - activemq::util::CMSExceptionSupport, 1078
- createNegotiator
 - activemq::wireformat::openwire::OpenWireFormat, 2703
 - activemq::wireformat::stomp::StompWireFormat, 3408
 - activemq::wireformat::WireFormat, 3714
- createObject
 - activemq::wireformat::openwire::marshal::DataStreamMarshal, 1505
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM, 176
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesI, 215
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapM, 334
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessa, 360
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObject, 404
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueue, 446
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStream, 505

activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	558	activemq::wireformat::openwire::marshal::v1::LocalTransactionMarshaller	2225
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	590	activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	2416
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	618	activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	2455
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMessageMarshaller	646	activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	2484
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	810	activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	2520
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	841	activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	2585
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	1186	activemq::wireformat::openwire::marshal::v1::NetworkBridgeMarshaller	2638
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	1217	activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	2753
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	1247	activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	2864
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	1277	activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	2895
activemq::wireformat::openwire::marshal::v1::ConsumerGroupMarshaller	1320	activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	2911
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	1347	activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	3004
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	1379	activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	3020
activemq::wireformat::openwire::marshal::v1::ContactCommandMarshaller	1406	activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	3051
activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller	1439	activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	3104
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	1501	activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	3186
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	1631	activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	3201
activemq::wireformat::openwire::marshal::v1::DiscardResponseMarshaller	1664	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	3261
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	1744	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	3443
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	1832	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	3604
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	1976	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	3745
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	2038	activemq::wireformat::openwire::marshal::v1::XATransactionInfoMarshaller	3782
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	2067	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	184
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	2089	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	231
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaller	2120	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	346
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	2146	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	372
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	2179	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	416

activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2130	activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	2130
458		activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	2167
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMessageMarshaller	2167	activemq::wireformat::openwire::marshal::v2::LocalTransactionMarshaller	2209
517		activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	2404
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueFormatMarshaller	2209	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2439
570		activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2472
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueFormatMarshaller	2404	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2500
598		activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller	2569
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueFormatMarshaller	2439	activemq::wireformat::openwire::marshal::v2::NetworkBridgeFormatMarshaller	2618
630		activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	2736
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueFormatMarshaller	2472	activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	2844
658		activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	2875
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2500	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	2907
822		activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	2992
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2569	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller	3028
853		activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	3055
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2618	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	3090
1198		activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	3166
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2736	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	3209
1205		activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	3257
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	2844	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	3459
1235		activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	3620
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	2875	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	3737
1265		activemq::wireformat::openwire::marshal::v2::XATransactionInfoMarshaller	3774
activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	2907	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller	172
1308		activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMarshaller	211
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	2992	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMarshaller	330
1335			
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	3028		
1367			
activemq::wireformat::openwire::marshal::v2::ContainerMarshaller	3055		
1394			
activemq::wireformat::openwire::marshal::v2::DataMarshaller	3090		
1427			
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	3166		
1489			
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	3209		
1619			
activemq::wireformat::openwire::marshal::v2::DiscardResponseMarshaller	3257		
1652			
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	3459		
1728			
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	3620		
1820			
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	3737		
1964			
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	3774		
2022			
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	172		
2051			
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	211		
2073			
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	330		
2104			

activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller;openwire::marshal::v3::JournalTraceMa	2077
356	
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller;marshal::v3::JournalTransact	2108
400	
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller;openwire::marshal::v3::KeepAliveInfoM	2134
442	
activemq::wireformat::openwire::marshal::v3::ActiveMQSequenceMessageMarshaller;marshal::v3::LastPartialComm	2163
501	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller;openwire::marshal::v3::LocalTransaction	2213
554	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller;openwire::marshal::v3::MessageAckMar	2408
582	
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller;openwire::marshal::v3::MessageDispatc	2443
610	
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller;openwire::marshal::v3::MessageDispatc	2476
638	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller;openwire::marshal::v3::MessageIdMarsh	2512
802	
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller;openwire::marshal::v3::MessagePullMar	2577
833	
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller;openwire::marshal::v3::NetworkBridgeF	2630
1178	
activemq::wireformat::openwire::marshal::v3::ConnectionErrorHandler;openwire::marshal::v3::PartialCommand	2745
1209	
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller;openwire::marshal::v3::ProducerAckMar	2852
1239	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller;openwire::marshal::v3::ProducerIdMar	2883
1269	
activemq::wireformat::openwire::marshal::v3::ConsumerGroupMarshaller;openwire::marshal::v3::ProducerInfoMa	2919
1312	
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller;openwire::marshal::v3::RemoveInfoMar	3000
1339	
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller;openwire::marshal::v3::RemoveSubscrip	3024
1371	
activemq::wireformat::openwire::marshal::v3::ContextCommandMarshaller;openwire::marshal::v3::ReplayCommand	3059
1398	
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller;openwire::marshal::v3::ResponseMarsha	3099
1431	
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller;openwire::marshal::v3::SessionIdMarsha	3182
1493	
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller;openwire::marshal::v3::SessionInfoMar	3205
1623	
activemq::wireformat::openwire::marshal::v3::DiscoveryInfoMarshaller;openwire::marshal::v3::ShutdownInfoMa	3269
1656	
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller;openwire::marshal::v3::SubscriptionInfo	3439
1732	
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller;openwire::marshal::v3::TransactionInfoM	3608
1824	
activemq::wireformat::openwire::marshal::v3::IntegrationResponseMarshaller;openwire::marshal::v3::WireFormatInfo	3749
1968	
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller;openwire::marshal::v3::XATransactionI	3786
2030	
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller;openwire::marshal::v4::ActiveMQBlobM	180
2055	

activemq::wireformat::openwire::marshal::v4::ActiveMQByteMessageMarshaller	219	2034	activemq::wireformat::openwire::marshal::v4::JournalQueueAck
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	338	2063	activemq::wireformat::openwire::marshal::v4::JournalTopicAck
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	364	2085	activemq::wireformat::openwire::marshal::v4::JournalTraceMa
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	408	2116	activemq::wireformat::openwire::marshal::v4::JournalTransact
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	450	2138	activemq::wireformat::openwire::marshal::v4::KeepAliveInfoM
activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceMessageMarshaller	509	2175	activemq::wireformat::openwire::marshal::v4::LastPartialComm
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	562	2221	activemq::wireformat::openwire::marshal::v4::LocalTransaction
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	586	2412	activemq::wireformat::openwire::marshal::v4::MessageAckMar
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	614	2451	activemq::wireformat::openwire::marshal::v4::MessageDispatc
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	642	2480	activemq::wireformat::openwire::marshal::v4::MessageDispatc
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	806	2504	activemq::wireformat::openwire::marshal::v4::MessageIdMarsh
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	837	2581	activemq::wireformat::openwire::marshal::v4::MessagePullMar
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	1182	2634	activemq::wireformat::openwire::marshal::v4::NetworkBridgeF
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	1213	2749	activemq::wireformat::openwire::marshal::v4::PartialCommand
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	1243	2848	activemq::wireformat::openwire::marshal::v4::ProducerAckMar
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1273	2879	activemq::wireformat::openwire::marshal::v4::ProducerIdMars
activemq::wireformat::openwire::marshal::v4::ConsumerGroupMarshaller	1316	2903	activemq::wireformat::openwire::marshal::v4::ProducerInfoMa
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	1343	3012	activemq::wireformat::openwire::marshal::v4::RemoveInfoMars
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1375	3040	activemq::wireformat::openwire::marshal::v4::RemoveSubscrip
activemq::wireformat::openwire::marshal::v4::ContentCommandMarshaller	1402	3047	activemq::wireformat::openwire::marshal::v4::ReplayCommand
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller	1435	3086	activemq::wireformat::openwire::marshal::v4::ResponseMarsha
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	1497	3170	activemq::wireformat::openwire::marshal::v4::SessionIdMarsha
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	1627	3213	activemq::wireformat::openwire::marshal::v4::SessionInfoMars
activemq::wireformat::openwire::marshal::v4::DiscardEventMarshaller	1660	3273	activemq::wireformat::openwire::marshal::v4::ShutdownInfoMa
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	1740	3451	activemq::wireformat::openwire::marshal::v4::SubscriptionInfo
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	1828	3616	activemq::wireformat::openwire::marshal::v4::TransactionInfo
activemq::wireformat::openwire::marshal::v4::IntegrationResponseMarshaller	1972	3741	activemq::wireformat::openwire::marshal::v4::WireFormatInfo

activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	1836
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	3778
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobWireFormatMarshaller	188
activemq::wireformat::openwire::marshal::v5::ActiveMQByteWireFormatMarshaller	1980
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalQueueAckMarshaller	223
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTopicAckMarshaller	2026
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTraceMarshaller	342
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2047
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTraceMarshaller	368
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2093
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	412
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2112
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	454
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2142
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	513
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2171
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	566
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2217
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	594
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2420
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	622
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2447
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	650
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2488
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	814
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2508
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	845
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2573
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1190
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2626
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1221
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2741
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1251
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2856
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1281
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2887
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1324
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	2915
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1351
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	3008
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1383
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	3036
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1410
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	3067
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1443
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	3095
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1481
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	3178
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1639
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	3197
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1668
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	3265
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	1736
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller	3447

activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	1648
3600	
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	1724
3729	
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	1816
3790	
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobWireFormatMessageMarshaller	1960
192	
activemq::wireformat::openwire::marshal::v6::ActiveMQBinaryWireFormatMessageMarshaller	2018
227	
activemq::wireformat::openwire::marshal::v6::ActiveMQMapWireFormatMessageMarshaller	2059
350	
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	2081
376	
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectWireFormatMessageMarshaller	2100
420	
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueWireFormatMessageMarshaller	2126
462	
activemq::wireformat::openwire::marshal::v6::ActiveMQSequenceWireFormatMessageMarshaller	2159
521	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueFormatMarshaller	2205
574	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	2400
602	
activemq::wireformat::openwire::marshal::v6::ActiveMQTextWireFormatMessageMarshaller	2459
626	
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicInfoMarshaller	2468
654	
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	2516
818	
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller	2589
849	
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller	2622
1194	
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller	2732
1225	
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller	2860
1255	
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller	2891
1285	
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller	2923
1328	
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	2996
1355	
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller	3032
1387	
activemq::wireformat::openwire::marshal::v6::ContainerCommandMarshaller	3063
1414	
activemq::wireformat::openwire::marshal::v6::DataActiveResponseMarshaller	3108
1447	
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller	3174
1485	
activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller	3193
1635	

- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshallerImplFactory, 3314
- 3253
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshallerImplFactory, 2773
- 3455
- activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3157
- 3612
- activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 283
- 3733
- activemq::wireformat::openwire::marshal::v6::XATransactionInfoMarshaller, 3770
- activemq::cmsutil::PooledSession, 2773
- activemq::core::ActiveMQSession, 474
- cms::Session, 3157
- createProducer
- activemq::cmsutil::PooledSession, 2772
- activemq::core::ActiveMQSession, 473
- cms::Session, 3156
- createQueryString
- activemq::util::URISupport, 3685
- createQueue
- activemq::cmsutil::PooledSession, 2773
- activemq::core::ActiveMQSession, 474
- cms::Session, 3156
- createRemoveCommand
- activemq::commands::ConnectionInfo, 1259
- activemq::commands::ConsumerInfo, 1360
- activemq::commands::ProducerInfo, 2899
- activemq::commands::SessionInfo, 3190
- createServerSocket
- decaf::internal::net::DefaultServerSocketFactory, 1575, 1576
- decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1584, 1585
- decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2671, 2672
- decaf::net::ServerSocketFactory, 3146, 3147
- createSession
- activemq::cmsutil::CmsAccessor, 1070
- activemq::core::ActiveMQConnection, 239
- cms::Connection, 1170
- createShortBuffer
- decaf::internal::nio::BufferFactory, 878, 879
- createSocket
- activemq::transport::tcp::SslTransport, 3349
- activemq::transport::tcp::TcpTransport, 3512
- decaf::internal::net::DefaultSocketFactory, 1579, 1580
- decaf::internal::net::ssl::DefaultSSLSocketFactory, 1589–1591
- decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2692–2694
- decaf::net::SocketFactory, 3302–3304
- decaf::net::ssl::SSLSocketFactory, 3346
- createSocketImpl
- activemq::transport::AbstractTransportFactory, 165
- activemq::wireformat::openwire::OpenWireFormatFactory, 2713
- activemq::wireformat::stomp::StompWireFormatFactory, 3411
- activemq::wireformat::WireFormatFactory, 3717
- criticalSection
- decaf::util::concurrent::ConditionHandle, 1163
- ct_data
- deflate.h, 4202
- ct_data_s, 1422
- code, 1423
- dad, 1423
- dl, 1423
- fc, 1423
- freq, 1423
- len, 1423
- CUNSUMER_MAXPENDINGMSGLIMIT
- activemq::core::ActiveMQConstants, 267
- currentThread
- decaf::lang::Thread, 3525
- currentTimeMillis
- decaf::lang::System, 3491
- d_buf
- internal_state, 1985

- d_code
 - deflate.h, 4202
- D_CODES
 - deflate.h, 4202
- d_desc
 - internal_state, 1985
- Dad
 - deflate.h, 4202
- dad
 - ct_data_s, 1423
- data
 - activemq::commands::DataArrayResponse, 1425
 - activemq::commands::DataResponse, 1479
 - activemq::commands::PartialCommand, 2731
- data_type
 - z_stream_s, 3795
- DataArrayResponse
 - activemq::commands::DataArrayResponse, 1424
- DataArrayResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1439
 - activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1427
 - activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1431
 - activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1435
 - activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1443
 - activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller, 1447
- DataFormatException
 - decaf::util::zip::DataFormatException, 1450, 1451
- DataInputStream
 - decaf::io::DataInputStream, 1462
- DataOutputStream
 - decaf::io::DataOutputStream, 1475
- DataResponse
 - activemq::commands::DataResponse, 1478
- DataResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1501
 - activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1489
 - activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1493
 - activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1497
 - activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1481
- activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 1485
- dataStructure
 - activemq::commands::Message, 2374
- Date
 - decaf::util::Date, 1560
- DAYS
 - decaf::util::concurrent::TimeUnit, 3567
- DEBUG
 - decaf::util::logging::Level, 2189
- Debug
 - decaf::util::logging, 140
- debug
 - decaf::util::logging::Logger, 2241
 - decaf::util::logging::SimpleLogger, 3280
- decaf, 120
- decaf/lang/exceptions/ExceptionDefines.h
 - DECAF_CATCH_EXCEPTION_-CONVERT, 3857
 - DECAF_CATCH_NOTHROW, 3857
 - DECAF_CATCH_RETHROW, 3857
 - DECAF_CATCHALL_NOTHROW, 3858
 - DECAF_CATCHALL_RETHROW, 3858
- decaf/util/Config.h
 - DECAF_APP_POOL, 3890
 - DECAF_APP_POOL_SIZE, 3890
 - HAVE_READSHARED, 3890
 - HAVE_UUID_T, 3890
 - HAVE_UUID_NAMESPACE, 3890
- decaf::internal, 121
- DataArrayResponseMarshaller, 668
- DataArrayResponseMarshaller, ~AprPool, 669
- AprPool, 669
- AprPoolResponseMarshaller, cleanup, 669
- getAprPool, 669
- getGlobalPool, 669
- decaf::internal::DecafRuntime, 1563
- ~DecafRuntime, 1563
- DecafRuntime, 1563
- getGlobalPool, 1564
- decaf::internal::io, 121
- decaf::internal::io::StandardErrorOutputStream, 3351
- ~StandardErrorOutputStream, 3352
- doWriteArrayBounded, 3352
- doWriteBy, 3352
- flush, 3352
- StandardErrorOutputStream, 3352
- decaf::internal::io::StandardInputStream, 3352
- StandardInputStream, 3353
- available, 3353
- doReadBy, 3353
- StandardInputStream, 3353

- decaf::internal::io::StandardOutputStream, 3354
 - ~StandardOutputStream, 3354
 - close, 3354
 - doWriteArrayBounded, 3354
 - doWriteByte, 3355
 - flush, 3355
 - StandardOutputStream, 3354
- decaf::internal::net, 122
- decaf::internal::net::DefaultServerSocketFactory, 1574
 - ~DefaultServerSocketFactory, 1575
 - createServerSocket, 1575, 1576
 - DefaultServerSocketFactory, 1575
- decaf::internal::net::DefaultSocketFactory, 1577
 - ~DefaultSocketFactory, 1579
 - createSocket, 1579, 1580
 - DefaultSocketFactory, 1579
- decaf::internal::net::Network, 2611
 - ~Network, 2612
 - addAsResource, 2612
 - addNetworkResource, 2612
 - getNetworkRuntime, 2613
 - getRuntimeLock, 2613
 - initializeNetworking, 2613
 - Network, 2612
 - shutdownNetworking, 2613
- decaf::internal::net::SocketFileDescriptor, 3305
 - ~SocketFileDescriptor, 3305
 - getValue, 3305
 - SocketFileDescriptor, 3305
- decaf::internal::net::ssl, 122
- decaf::internal::net::ssl::DefaultSSLContext, 1581
 - ~DefaultSSLContext, 1582
 - DefaultSSLContext, 1582
 - getContext, 1582
- decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1582
 - ~DefaultSSLServerSocketFactory, 1584
 - createServerSocket, 1584, 1585
 - DefaultSSLServerSocketFactory, 1584
 - getDefaultCipherSuites, 1586
 - getSupportedCipherSuites, 1586
- decaf::internal::net::ssl::DefaultSSLSocketFactory, 1587
 - ~DefaultSSLSocketFactory, 1589
 - createSocket, 1589–1591
 - DefaultSSLSocketFactory, 1589
 - getDefaultCipherSuites, 1592
 - getSupportedCipherSuites, 1592
- decaf::internal::net::ssl::openssl, 123
- decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2658
 - ~OpenSSLContextSpi, 2660
 - OpenSSLContextSpi, 2660
 - OpenSSLSocket, 2661
 - OpenSSLSocketFactory, 2661
 - providerGetServerSocketFactory, 2660
 - providerGetSocketFactory, 2660
 - providerInit, 2660
- decaf::internal::net::ssl::openssl::OpenSSLParameters, 2661
 - ~OpenSSLParameters, 2662
 - clone, 2662
 - getEnabledCipherSuites, 2662
 - getEnabledProtocols, 2663
 - getNeedClientAuth, 2663
 - getSupportedCipherSuites, 2663
 - getSupportedProtocols, 2663
 - getUseClientMode, 2663
 - getWantClientAuth, 2663
 - setEnabledCipherSuites, 2663
 - setEnabledProtocols, 2663
 - setNeedClientAuth, 2663
 - setUseClientMode, 2663
 - setWantClientAuth, 2663
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2664
 - ~OpenSSLServerSocket, 2666
 - accept, 2666
 - getEnabledCipherSuites, 2666
 - getEnabledProtocols, 2666
 - getNeedClientAuth, 2667
 - getSupportedCipherSuites, 2667
 - getSupportedProtocols, 2667
 - getWantClientAuth, 2667
 - OpenSSLServerSocket, 2666
 - setEnabledCipherSuites, 2668
 - setEnabledProtocols, 2668
 - setNeedClientAuth, 2668
 - setWantClientAuth, 2669
- decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2669
 - ~OpenSSLServerSocketFactory, 2671
 - createServerSocket, 2671, 2672
 - getDefaultCipherSuites, 2673
 - getSupportedCipherSuites, 2673
 - OpenSSLServerSocketFactory, 2671
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2673
 - ~OpenSSLSocket, 2678
 - available, 2678
 - close, 2679
 - connect, 2679
 - getEnabledCipherSuites, 2679
 - getEnabledProtocols, 2679
 - getInputStream, 2680

- getNeedClientAuth, 2680
- getOutputStream, 2680
- getSupportedCipherSuites, 2681
- getSupportedProtocols, 2681
- getUseClientMode, 2681
- getWantClientAuth, 2682
- OpenSSLSocket, 2678
- read, 2682
- sendUrgentData, 2682
- setEnabledCipherSuites, 2683
- setEnabledProtocols, 2683
- setNeedClientAuth, 2683
- setOOBInline, 2684
- setUseClientMode, 2684
- setWantClientAuth, 2684
- shutdownInput, 2685
- shutdownOutput, 2685
- startHandshake, 2685
- write, 2685
- decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2686
 - ~OpenSSLSocketException, 2689
 - clone, 2689
 - getErrorString, 2689
 - OpenSSLSocketException, 2687, 2688
- decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2689
 - ~OpenSSLSocketFactory, 2692
 - createSocket, 2692–2694
 - getDefaultCipherSuites, 2695
 - getSupportedCipherSuites, 2695
 - OpenSSLSocketFactory, 2692
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2696
 - ~OpenSSLSocketInputStream, 2697
 - available, 2697
 - close, 2697
 - doReadArrayBounded, 2697
 - doReadByte, 2697
 - OpenSSLSocketInputStream, 2697
 - skip, 2698
- decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2698
 - ~OpenSSLSocketOutputStream, 2699
 - close, 2699
 - doWriteArrayBounded, 2699
 - doWriteByte, 2699
 - OpenSSLSocketOutputStream, 2699
- decaf::internal::net::tcp, 123
- decaf::internal::net::tcp::TcpSocket, 3497
 - ~TcpSocket, 3500
 - accept, 3501
 - available, 3501
 - bind, 3501
 - checkResult, 3501
 - close, 3501
 - connect, 3501
 - create, 3502
 - getInputStream, 3502
 - getLocalAddress, 3502
 - getOption, 3503
 - getOutputStream, 3503
 - getSocketHandle, 3503
 - isClosed, 3503
 - isConnected, 3504
 - listen, 3504
 - read, 3504
 - setOption, 3504
 - shutdownInput, 3505
 - shutdownOutput, 3505
 - TcpSocket, 3500
 - write, 3505
- decaf::internal::net::tcp::TcpSocketInputStream, 3506
 - ~TcpSocketInputStream, 3507
 - available, 3507
 - close, 3507
 - doReadArrayBounded, 3508
 - doReadByte, 3508
 - skip, 3508
 - TcpSocketInputStream, 3507
- decaf::internal::net::tcp::TcpSocketOutputStream, 3509
 - ~TcpSocketOutputStream, 3509
 - close, 3510
 - doWriteArrayBounded, 3510
 - doWriteByte, 3510
 - TcpSocketOutputStream, 3509
- decaf::internal::net::URIEncoderDecoder, 3672
 - ~URIEncoderDecoder, 3673
 - decode, 3673
 - encodeOthers, 3673
 - quoteIllegal, 3673
 - URIEncoderDecoder, 3673
 - validate, 3674
- URIHelperSimple, 3674
- decaf::internal::net::URIHelper, 3674
 - ~URIHelper, 3676
 - isValidDomainName, 3676
 - isValidHexChar, 3677
 - isValidHost, 3677
 - isValidIP4Word, 3677
 - isValidIP6Address, 3677
 - isValidIPv4Address, 3678
 - parseAuthority, 3678
 - parseURI, 3678
 - URIHelper, 3676
 - validateAuthority, 3679

- validateFragment, 3679
- validatePath, 3679
- validateQuery, 3680
- validateScheme, 3680
- validateSsp, 3680
- validateUserinfo, 3681
- decaf::internal::net::URIType, 3690
 - ~URIType, 3692
 - getAuthority, 3692
 - getFragment, 3692
 - getHost, 3693
 - getPath, 3693
 - getPort, 3693
 - getQuery, 3693
 - getScheme, 3693
 - getSchemeSpecificPart, 3693
 - getSource, 3694
 - getUserInfo, 3694
 - isAbsolute, 3694
 - isOpaque, 3694
 - isServerAuthority, 3694
 - isValid, 3694
 - setAbsolute, 3695
 - setAuthority, 3695
 - setFragment, 3695
 - setHost, 3695
 - setOpaque, 3695
 - setPath, 3695
 - setPort, 3696
 - setQuery, 3696
 - setScheme, 3696
 - setSchemeSpecificPart, 3696
 - setServerAuthority, 3696
 - setSource, 3697
 - setUserInfo, 3697
 - setValid, 3697
 - URIType, 3692
- decaf::internal::nio, 124
- decaf::internal::nio::BufferFactory, 869
 - ~BufferFactory, 871
 - createByteBuffer, 871, 872
 - createCharBuffer, 872, 873
 - createDoubleBuffer, 873, 874
 - createFloatBuffer, 875
 - createIntBuffer, 876, 877
 - createLongBuffer, 877, 878
 - createShortBuffer, 878, 879
- decaf::internal::nio::ByteBuffer, 915
 - ~ByteBuffer, 927
 - array, 928
 - arrayOffset, 928
 - asCharBuffer, 928
 - asDoubleBuffer, 929
 - asFloatBuffer, 929
 - asIntBuffer, 929
 - asLongBuffer, 930
 - asReadOnlyBuffer, 930
 - asShortBuffer, 931
 - ByteBuffer, 926, 927
 - compact, 931
 - duplicate, 931
 - get, 932
 - getChar, 932, 933
 - getDouble, 933
 - getFloat, 934
 - getInt, 934, 935
 - getLong, 935
 - getShort, 936
 - hasArray, 936
 - isReadOnly, 937
 - put, 937
 - putChar, 938
 - putDouble, 939
 - putFloat, 939, 940
 - putInt, 940, 941
 - putLong, 941, 942
 - putShort, 942
 - setReadOnly, 943
 - slice, 943
- decaf::internal::nio::CharArrayBuffer, 1027
 - ~CharArrayBuffer, 1032
 - _array, 1037
 - array, 1032
 - arrayOffset, 1032
 - asReadOnlyBuffer, 1033
 - CharArrayBuffer, 1031, 1032
 - compact, 1033
 - duplicate, 1034
 - get, 1034
 - hasArray, 1035
 - isReadOnly, 1035
 - length, 1037
 - offset, 1037
 - put, 1035
 - readOnly, 1037
 - setReadOnly, 1036
 - slice, 1036
 - subSequence, 1036
- decaf::internal::nio::DoubleArrayBuffer, 1683
 - ~DoubleArrayBuffer, 1688
 - array, 1688
 - arrayOffset, 1688
 - asReadOnlyBuffer, 1689
 - compact, 1689
 - DoubleArrayBuffer, 1686, 1687
 - duplicate, 1689
 - get, 1690
 - hasArray, 1690

- isReadOnly, 1691
- put, 1691
- setReadOnly, 1692
- slice, 1692
- decaf::internal::nio::FloatArrayBuffer, 1791
 - ~FloatArrayBuffer, 1795
 - array, 1796
 - arrayOffset, 1796
 - asReadOnlyBuffer, 1796
 - compact, 1797
 - duplicate, 1797
 - FloatArrayBuffer, 1794, 1795
 - get, 1797, 1798
 - hasArray, 1798
 - isReadOnly, 1798
 - put, 1799
 - setReadOnly, 1799
 - slice, 1799
- decaf::internal::nio::IntArrayBuffer, 1921
 - ~IntArrayBuffer, 1926
 - array, 1926
 - arrayOffset, 1926
 - asReadOnlyBuffer, 1927
 - compact, 1927
 - duplicate, 1928
 - get, 1928
 - hasArray, 1929
 - IntArrayBuffer, 1925, 1926
 - isReadOnly, 1929
 - put, 1929
 - setReadOnly, 1930
 - slice, 1930
- decaf::internal::nio::LongArrayBuffer, 2281
 - ~LongArrayBuffer, 2286
 - array, 2286
 - arrayOffset, 2287
 - asReadOnlyBuffer, 2287
 - compact, 2287
 - duplicate, 2288
 - get, 2288
 - hasArray, 2289
 - isReadOnly, 2289
 - LongArrayBuffer, 2285, 2286
 - put, 2289, 2290
 - setReadOnly, 2290
 - slice, 2290
- decaf::internal::nio::ShortArrayBuffer, 3229
 - ~ShortArrayBuffer, 3234
 - array, 3234
 - arrayOffset, 3234
 - asReadOnlyBuffer, 3235
 - compact, 3235
 - duplicate, 3235
 - get, 3236
 - hasArray, 3236
 - isReadOnly, 3237
 - put, 3237
 - setReadOnly, 3238
 - ShortArrayBuffer, 3232, 3233
 - slice, 3238
- decaf::internal::security, 124
- decaf::internal::security::SecureRandomImpl, 3121
 - ~SecureRandomImpl, 3122
 - providerGenerateSeed, 3122, 3123
 - providerNextBytes, 3123
 - providerSetSeed, 3123, 3124
 - SecureRandomImpl, 3122
- decaf::internal::util, 124
- decaf::internal::util::ByteArrayAdapter, 893
 - ~ByteArrayAdapter, 900
 - ByteArrayAdapter, 898–900
 - clear, 901
 - get, 901
 - getByteArray, 901
 - getCapacity, 901
 - getChar, 901
 - getCharArray, 902
 - getCharCapacity, 902
 - getDouble, 902
 - getDoubleArray, 903
 - getDoubleAt, 903
 - getDoubleCapacity, 903
 - getFloat, 903
 - getFloatArray, 904
 - getFloatAt, 904
 - getFloatCapacity, 904
 - getInt, 905
 - getIntArray, 905
 - getIntAt, 905
 - getIntCapacity, 906
 - getLong, 906
 - getLongArray, 906
 - getLongAt, 906
 - getLongCapacity, 907
 - getShort, 907
 - getShortArray, 907
 - getShortAt, 908
 - getShortCapacity, 908
 - operator[], 908
 - put, 909
 - putChar, 909
 - putDouble, 909
 - putDoubleAt, 910
 - putFloat, 910
 - putFloatAt, 911
 - putInt, 911
 - putIntAt, 911

- putLong, 912
- putLongAt, 912
- putShort, 913
- putShortAt, 913
- read, 913
- resize, 914
- write, 914
- decaf::internal::util::concurrent, 125
- decaf::internal::util::concurrent::ConditionImpl, 1163
 - create, 1164
 - destroy, 1164
 - notify, 1164
 - notifyAll, 1165
 - wait, 1165
- decaf::internal::util::concurrent::MutexImpl, 2609
 - create, 2610
 - destroy, 2610
 - lock, 2610
 - trylock, 2610
 - unlock, 2611
- decaf::internal::util::concurrent::SynchronizableImpl, 3473
 - ~SynchronizableImpl, 3474
 - lock, 3474
 - notify, 3474
 - notifyAll, 3474
 - SynchronizableImpl, 3474
 - tryLock, 3475
 - unlock, 3475
 - wait, 3475, 3476
- decaf::internal::util::concurrent::Transferer, 3625
- decaf::internal::util::concurrent::TransferQueue, 3625
 - ~TransferQueue, 3626
 - transfer, 3626
 - TransferQueue, 3626
- decaf::internal::util::concurrent::TransferStack, 3627
 - ~TransferStack, 3628
 - transfer, 3628
 - TransferStack, 3628
- decaf::internal::util::GenericResource, 1847
 - ~GenericResource, 1848
 - GenericResource, 1848
 - getManaged, 1848
 - setManaged, 1848
- decaf::internal::util::HexStringParser, 1856
 - ~HexStringParser, 1856
 - HexStringParser, 1856
 - parse, 1856
 - parseDouble, 1856
 - parseFloat, 1857
- decaf::internal::util::Resource, 3072
 - ~Resource, 3073
- decaf::internal::util::ResourceLifecycleManager, 3073
 - ~ResourceLifecycleManager, 3073
 - addResource, 3073
 - destroyResources, 3073
 - ResourceLifecycleManager, 3073
- decaf::internal::util::TimerTaskHeap, 3556
 - ~TimerTaskHeap, 3557
 - adjustMinimum, 3557
 - decaf::util::TimerTask, 3556
 - deleteIfCancelled, 3557
 - find, 3557
 - insert, 3558
 - isEmpty, 3558
 - peek, 3558
 - remove, 3558
 - reset, 3558
 - size, 3558
 - TimerTaskHeap, 3557
- decaf::io, 125
- decaf::io::BlockingByteArrayInputStream, 771
 - ~BlockingByteArrayInputStream, 772
 - available, 773
 - BlockingByteArrayInputStream, 772
 - close, 773
 - doReadArrayBounded, 773
 - doReadByte, 773
 - setByteArray, 773
 - skip, 774
- decaf::io::BufferedInputStream, 861
 - ~BufferedInputStream, 864
 - available, 864
 - BufferedInputStream, 863
 - close, 864
 - doReadArrayBounded, 864
 - doReadByte, 865
 - mark, 865
 - markSupported, 865
 - reset, 865
 - skip, 866
- decaf::io::BufferedOutputStream, 867
 - ~BufferedOutputStream, 868
 - BufferedOutputStream, 867
 - doWriteArray, 868
 - doWriteArrayBounded, 868
 - doWriteByte, 868
 - flush, 868
- decaf::io::ByteArrayInputStream, 944
 - ~ByteArrayInputStream, 947
 - available, 947
 - ByteArrayInputStream, 946, 947

- doReadArrayBounded, 948
- doReadByte, 948
- mark, 948
- markSupported, 948
- reset, 949
- setByteArray, 949, 950
- skip, 950
- decaf::io::ByteArrayOutputStream, 951
 - ~ByteArrayOutputStream, 952
 - ByteArrayOutputStream, 952
 - doWriteArrayBounded, 952
 - doWriteByte, 952
 - reset, 952
 - size, 953
 - toByteArray, 953
 - toString, 953
 - writeTo, 953
- decaf::io::Closeable, 1065
 - ~Closeable, 1066
 - close, 1066
- decaf::io::DataInput, 1452
 - ~DataInput, 1454
 - readBoolean, 1454
 - readByte, 1454
 - readChar, 1454
 - readDouble, 1455
 - readFloat, 1455
 - readFully, 1455, 1456
 - readInt, 1457
 - readLine, 1457
 - readLong, 1457
 - readShort, 1458
 - readString, 1458
 - readUnsignedByte, 1458
 - readUnsignedShort, 1459
 - readUTF, 1459
 - skipBytes, 1459
- decaf::io::DataInputStream, 1460
 - ~DataInputStream, 1462
 - DataInputStream, 1462
 - readBoolean, 1462
 - readByte, 1462
 - readChar, 1463
 - readDouble, 1463
 - readFloat, 1463
 - readFully, 1464
 - readInt, 1465
 - readLine, 1465
 - readLong, 1466
 - readShort, 1466
 - readString, 1466
 - readUnsignedByte, 1467
 - readUnsignedShort, 1467
 - readUTF, 1467
 - skipBytes, 1468
- decaf::io::DataOutput, 1468
 - ~DataOutput, 1470
 - writeBoolean, 1470
 - writeByte, 1470
 - writeBytes, 1470
 - writeChar, 1470
 - writeChars, 1471
 - writeDouble, 1471
 - writeFloat, 1471
 - writeInt, 1472
 - writeLong, 1472
 - writeShort, 1472
 - writeUnsignedShort, 1472
 - writeUTF, 1473
- decaf::io::DataOutputStream, 1473
 - ~DataOutputStream, 1475
 - buffer, 1476
 - DataOutputStream, 1475
 - doWriteArrayBounded, 1475
 - doWriteByte, 1475
 - size, 1475
 - writeBoolean, 1475
 - writeByte, 1476
 - writeBytes, 1476
 - writeChar, 1476
 - writeChars, 1476
 - writeDouble, 1476
 - writeFloat, 1476
 - writeInt, 1476
 - writeLong, 1476
 - writeShort, 1476
 - writeUnsignedShort, 1476
 - writeUTF, 1476
 - written, 1476
- decaf::io::EOFException, 1707
 - ~EOFException, 1709
 - clone, 1709
 - EOFException, 1708, 1709
- decaf::io::FileDescriptor, 1768
 - ~FileDescriptor, 1769
 - descriptor, 1769
 - err, 1769
 - FileDescriptor, 1769
 - in, 1769
 - out, 1769
 - readonly, 1770
 - sync, 1769
 - valid, 1769
- decaf::io::FilterInputStream, 1771
 - ~FilterInputStream, 1773
 - available, 1773
 - close, 1774
 - closed, 1777

- doReadArray, 1774
- doReadArrayBounded, 1774
- doReadByte, 1774
- FilterInputStream, 1773
- inputStream, 1777
- isClosed, 1774
- mark, 1775
- markSupported, 1775
- own, 1777
- reset, 1775
- skip, 1776
- decaf::io::FilterOutputStream, 1777
 - ~FilterOutputStream, 1778
 - close, 1778
 - closed, 1780
 - doWriteArray, 1778
 - doWriteArrayBounded, 1779
 - doWriteByte, 1779
 - FilterOutputStream, 1778
 - flush, 1779
 - isClosed, 1779
 - outputStream, 1780
 - own, 1780
 - toString, 1779
- decaf::io::Flushable, 1811
 - ~Flushable, 1811
 - flush, 1811
- decaf::io::InputStream, 1909
 - ~InputStream, 1911
 - available, 1911
 - close, 1912
 - doReadArray, 1912
 - doReadArrayBounded, 1912
 - doReadByte, 1912
 - InputStream, 1911
 - lock, 1913
 - mark, 1913
 - markSupported, 1913
 - notify, 1913
 - notifyAll, 1914
 - read, 1914, 1915
 - reset, 1916
 - skip, 1916
 - toString, 1917
 - tryLock, 1917
 - unlock, 1918
 - wait, 1918
- decaf::io::InputStreamReader, 1919
 - ~InputStreamReader, 1920
 - checkClosed, 1921
 - close, 1921
 - doReadArrayBounded, 1921
 - InputStreamReader, 1920
 - ready, 1921
- decaf::io::InterruptedIOException, 1990
 - ~InterruptedIOException, 1992
 - clone, 1992
 - InterruptedIOException, 1990, 1991
- decaf::io::IOException, 2003
 - ~IOException, 2005
 - clone, 2005
 - IOException, 2003, 2004
- decaf::io::OutputStream, 2718
 - ~OutputStream, 2720
 - close, 2720
 - doWriteArray, 2720
 - doWriteArrayBounded, 2721
 - doWriteByte, 2721
 - flush, 2721
 - lock, 2721
 - notify, 2722
 - notifyAll, 2722
 - OutputStream, 2720
 - toString, 2722
 - tryLock, 2722
 - unlock, 2723
 - wait, 2723, 2724
 - write, 2724, 2725
- decaf::io::OutputStreamWriter, 2726
 - ~OutputStreamWriter, 2727
 - checkClosed, 2727
 - close, 2727
 - doWriteArrayBounded, 2727
 - flush, 2727
 - OutputStreamWriter, 2727
- decaf::io::PushbackInputStream, 2940
 - ~PushbackInputStream, 2943
 - available, 2943
 - doReadArrayBounded, 2943
 - doReadByte, 2943
 - mark, 2943
 - markSupported, 2944
 - PushbackInputStream, 2942
 - reset, 2944
 - skip, 2945
 - unread, 2945, 2946
- decaf::io::Reader, 2960
 - ~Reader, 2962
 - doReadArray, 2962
 - doReadArrayBounded, 2962
 - doReadChar, 2962
 - doReadCharBuffer, 2962
 - doReadVector, 2962
 - mark, 2962
 - markSupported, 2963
 - read, 2963–2965
 - Reader, 2962
 - ready, 2965

- reset, 2965
- skip, 2966
- decaf::io::UnsupportedEncodingException, 3654
 - ~UnsupportedEncodingException, 3656
 - clone, 3656
 - UnsupportedEncodingException, 3655, 3656
- decaf::io::UTFDataFormatException, 3703
 - ~UTFDataFormatException, 3705
 - clone, 3705
 - UTFDataFormatException, 3704, 3705
- decaf::io::Writer, 3756
 - ~Writer, 3757
 - append, 3757, 3758
 - doAppendChar, 3758
 - doAppendCharSequence, 3759
 - doAppendCharSequenceStartEnd, 3759
 - doWriteArray, 3759
 - doWriteArrayBounded, 3759
 - doWriteChar, 3759
 - doWriteString, 3759
 - doWriteStringBounded, 3759
 - doWriteVector, 3759
 - write, 3759–3761
 - Writer, 3757
- decaf::lang, 127
 - operator==, 129, 130
- decaf::lang::Appendable, 666
 - ~Appendable, 667
 - append, 667
- decaf::lang::ArrayPointer, 669
 - ~ArrayPointer, 673
 - ArrayPointer, 672, 673
 - clone, 673
 - ConstReferenceType, 672
 - CounterType, 672
 - get, 673
 - length, 673
 - operator=, 674
 - operator==, 674, 676
 - operator[], 674, 675
 - PointerType, 672
 - ReferenceType, 672
 - release, 675
 - reset, 675
 - swap, 675
- decaf::lang::ArrayPointerComparator, 676
 - compare, 677
 - operator(), 677
- decaf::lang::Boolean, 781
 - ~Boolean, 782
 - _FALSE, 785
 - _TRUE, 785
- Boolean, 782
- booleanValue, 782
- compareTo, 782
- equals, 783
- operator<, 783
- operator==, 784
- parseBoolean, 784
- toString, 784, 785
- valueOf, 785
- decaf::lang::Byte, 884
 - ~Byte, 887
 - Byte, 886
 - byteValue, 887
 - compareTo, 887
 - decode, 887
 - doubleValue, 888
 - equals, 888
 - floatValue, 888
 - intValue, 888
 - longValue, 889
 - MAX_VALUE, 893
 - MIN_VALUE, 893
 - operator<, 889
 - operator==, 889, 890
 - parseByte, 890
 - shortValue, 891
 - SIZE, 893
 - toString, 891
 - valueOf, 891, 892
- decaf::lang::Character, 1019
 - byteValue, 1022
 - Character, 1021
 - compareTo, 1022
 - digit, 1022
 - doubleValue, 1023
 - equals, 1023
 - floatValue, 1023
 - intValue, 1023
 - isDigit, 1023
 - isISOControl, 1024
 - isLetter, 1024
 - isLetterOrDigit, 1024
 - isLowerCase, 1024
 - isUpperCase, 1024
 - isWhitespace, 1024
 - longValue, 1024
 - MAX_RADIX, 1026
 - MAX_VALUE, 1026
 - MIN_RADIX, 1026
 - MIN_VALUE, 1026
 - operator<, 1024, 1025
 - operator==, 1025
 - shortValue, 1026
 - SIZE, 1026

- toString, 1026
- valueOf, 1026
- decaf::lang::CharSequence, 1053
 - ~CharSequence, 1054
 - charAt, 1054
 - length, 1054
 - subSequence, 1054
 - toString, 1055
- decaf::lang::Comparable, 1125
 - ~Comparable, 1126
 - compareTo, 1126
 - equals, 1126
 - operator<, 1126
 - operator==, 1127
- decaf::lang::Double, 1672
 - ~Double, 1674
 - byteValue, 1675
 - compare, 1675
 - compareTo, 1675
 - Double, 1674
 - doubleToLongBits, 1676
 - doubleToRawLongBits, 1676
 - doubleValue, 1677
 - equals, 1677
 - floatValue, 1677
 - intValue, 1677
 - isInfinite, 1678
 - isNaN, 1678
 - longBitsToDouble, 1678
 - longValue, 1679
 - MAX_VALUE, 1682
 - MIN_VALUE, 1682
 - NaN, 1682
 - NEGATIVE_INFINITY, 1683
 - operator<, 1679
 - operator==, 1679, 1680
 - parseDouble, 1680
 - POSITIVE_INFINITY, 1683
 - shortValue, 1680
 - SIZE, 1683
 - toHexString, 1680
 - toString, 1681
 - valueOf, 1682
- decaf::lang::DYNAMIC_CAST_TOKEN, 1704
- decaf::lang::Exception, 1712
 - ~Exception, 1715
 - buildMessage, 1715
 - cause, 1718
 - clone, 1715
 - Exception, 1713, 1714
 - getCause, 1716
 - getMessage, 1716
 - getStackTrace, 1716
 - getStackTraceString, 1717
 - initCause, 1717
 - message, 1718
 - operator=, 1717
 - printStackTrace, 1717
 - setMark, 1717
 - setMessage, 1718
 - setStackTrace, 1718
 - stackTrace, 1718
 - what, 1718
- decaf::lang::exceptions, 130
- decaf::lang::exceptions::ClassCastException, 1062
 - ~ClassCastException, 1064
 - ClassCastException, 1063
 - clone, 1064
- decaf::lang::exceptions::IllegalArgumentException, 1863
 - ~IllegalArgumentException, 1865
 - clone, 1865
 - IllegalArgumentException, 1864
- decaf::lang::exceptions::IllegalMonitorStateException, 1865
 - ~IllegalMonitorStateException, 1867
 - clone, 1867
 - IllegalMonitorStateException, 1866, 1867
- decaf::lang::exceptions::IllegalStateException, 1869
 - ~IllegalStateException, 1871
 - clone, 1871
 - IllegalStateException, 1870, 1871
- decaf::lang::exceptions::IllegalThreadStateException, 1872
 - ~IllegalThreadStateException, 1873
 - clone, 1874
 - IllegalThreadStateException, 1872, 1873
- decaf::lang::exceptions::IndexOutOfBoundsException, 1877
 - ~IndexOutOfBoundsException, 1879
 - clone, 1879
 - IndexOutOfBoundsException, 1878, 1879
- decaf::lang::exceptions::InterruptedException, 1987
 - ~InterruptedException, 1989
 - clone, 1989
 - InterruptedException, 1988, 1989
- decaf::lang::exceptions::InvalidStateException, 2000
 - ~InvalidStateException, 2002
 - clone, 2002
 - InvalidStateException, 2001, 2002
- decaf::lang::exceptions::NoSuchElementException, 2645
 - ~NoSuchElementException, 2647
 - clone, 2647

- NoSuchElementException, 2646, 2647
- decaf::lang::exceptions::NullPointerException, 2650
 - ~NullPointerException, 2652
 - clone, 2652
 - NullPointerException, 2651, 2652
- decaf::lang::exceptions::NumberFormatException, 2655
 - ~NumberFormatException, 2657
 - clone, 2657
 - NumberFormatException, 2656, 2657
- decaf::lang::exceptions::RuntimeException, 3114
 - ~RuntimeException, 3116
 - clone, 3116
 - RuntimeException, 3115
- decaf::lang::exceptions::UnsupportedOperationException, 3657
 - ~UnsupportedOperationException, 3659
 - clone, 3659
 - UnsupportedOperationException, 3657, 3658
- decaf::lang::Float, 1780
 - ~Float, 1783
 - byteValue, 1783
 - compare, 1783
 - compareTo, 1783, 1784
 - doubleValue, 1784
 - equals, 1784
 - Float, 1782, 1783
 - floatToIntBits, 1784
 - floatToRawIntBits, 1785
 - floatValue, 1785
 - intBitsToFloat, 1785
 - intValue, 1786
 - isInfinite, 1786
 - isNaN, 1786
 - longValue, 1787
 - MAX_VALUE, 1790
 - MIN_VALUE, 1790
 - NaN, 1790
 - NEGATIVE_INFINITY, 1791
 - operator<, 1787
 - operator==, 1787, 1788
 - parseFloat, 1788
 - POSITIVE_INFINITY, 1791
 - shortValue, 1788
 - SIZE, 1791
 - toHexString, 1788
 - toString, 1789
 - valueOf, 1790
- decaf::lang::Integer, 1941
 - ~Integer, 1945
 - bitCount, 1945
 - byteValue, 1945
 - compareTo, 1945
 - decode, 1946
 - doubleValue, 1946
 - equals, 1946, 1947
 - floatValue, 1947
 - highestOneBit, 1947
 - Integer, 1944
 - intValue, 1947
 - longValue, 1947
 - lowestOneBit, 1948
 - MAX_VALUE, 1956
 - MIN_VALUE, 1956
 - numberOfLeadingZeros, 1948
 - numberOfTrailingZeros, 1948
 - operator<, 1949
 - operator==, 1949
 - parseInt, 1950
 - reverse, 1951
 - reverseBytes, 1951
 - rotateLeft, 1951
 - rotateRight, 1952
 - shortValue, 1952
 - signum, 1952
 - SIZE, 1956
 - toBinaryString, 1952
 - toHexString, 1953
 - toOctalString, 1953
 - toString, 1954
 - valueOf, 1954, 1955
- decaf::lang::Iterable, 2011
 - ~Iterable, 2012
 - iterator, 2012
- decaf::lang::Long, 2267
 - ~Long, 2270
 - bitCount, 2271
 - byteValue, 2271
 - compareTo, 2271
 - decode, 2272
 - doubleValue, 2272
 - equals, 2272
 - floatValue, 2273
 - highestOneBit, 2273
 - intValue, 2273
 - Long, 2270
 - longValue, 2273
 - lowestOneBit, 2273
 - MAX_VALUE, 2281
 - MIN_VALUE, 2281
 - numberOfLeadingZeros, 2274
 - numberOfTrailingZeros, 2274
 - operator<, 2275
 - operator==, 2275
 - parseLong, 2276

- reverse, 2277
- reverseBytes, 2277
- rotateLeft, 2277
- rotateRight, 2277
- shortValue, 2278
- signum, 2278
- SIZE, 2281
- toBinaryString, 2278
- toHexString, 2279
- toOctalString, 2279
- toString, 2279, 2280
- valueOf, 2280, 2281
- decaf::lang::Math, 2339
 - ~Math, 2341
 - abs, 2341, 2342
 - ceil, 2342
 - E, 2354
 - floor, 2344
 - Math, 2341
 - max, 2344, 2345
 - min, 2346, 2347
 - PI, 2354
 - pow, 2348
 - random, 2348
 - round, 2349
 - signum, 2349, 2350
 - sqrt, 2351
 - toDegrees, 2354
 - toRadians, 2354
- decaf::lang::Number, 2653
 - ~Number, 2654
 - byteValue, 2654
 - doubleValue, 2654
 - floatValue, 2654
 - intValue, 2654
 - longValue, 2654
 - shortValue, 2655
- decaf::lang::Pointer, 2756
 - ~Pointer, 2760
 - CounterType, 2758
 - dynamicCast, 2760
 - get, 2760
 - operator*, 2761
 - operator->, 2761
 - operator=, 2762
 - operator==, 2762, 2763
 - Pointer, 2758, 2759
 - PointerType, 2758
 - ReferenceType, 2758
 - release, 2762
 - reset, 2762
 - staticCast, 2763
 - swap, 2763
- decaf::lang::PointerComparator, 2764
 - compare, 2764
 - operator(), 2764
- decaf::lang::Readable, 2958
 - ~Readable, 2958
 - read, 2958
- decaf::lang::Runnable, 3111
 - ~Runnable, 3112
 - run, 3112
- decaf::lang::Runtime, 3112
 - ~Runtime, 3113
 - decaf::lang::System, 3494
 - decaf::lang::Thread, 3529
 - decaf::util::logging::LogManager, 2261
 - getRuntime, 3113
 - initializeRuntime, 3113
 - shutdownRuntime, 3113
- decaf::lang::Short, 3220
 - ~Short, 3223
 - byteValue, 3223
 - compareTo, 3223
 - decode, 3223
 - doubleValue, 3224
 - equals, 3224
 - floatValue, 3224
 - intValue, 3224
 - longValue, 3225
 - MAX_VALUE, 3229
 - MIN_VALUE, 3229
 - operator<, 3225
 - operator==, 3225, 3226
 - parseShort, 3226
 - reverseBytes, 3227
 - Short, 3222
 - shortValue, 3227
 - SIZE, 3229
 - toString, 3227
 - valueOf, 3228
- decaf::lang::STATIC_CAST_TOKEN, 3356
- decaf::lang::String, 3427
 - ~String, 3429
 - charAt, 3429
 - isEmpty, 3429
 - length, 3429
 - String, 3429
 - subSequence, 3429
 - toString, 3430
- decaf::lang::System, 3487
 - ~System, 3489
 - arraycopy, 3489, 3490
 - availableProcessors, 3490
 - clearProperty, 3491
 - currentTimeMillis, 3491
 - decaf::lang::Runtime, 3494
 - getenv, 3491, 3492

- getProperties, 3492
- getProperty, 3492
- nanoTime, 3493
- setenv, 3493
- setProperty, 3493
- System, 3489
- unsetenv, 3494
- decaf::lang::Thread, 3520
 - ~Thread, 3525
 - BLOCKED, 3523
 - currentThread, 3525
 - decaf::lang::Runtime, 3529
 - decaf::util::concurrent::locks::LockSupport, 3529
 - getId, 3525
 - getName, 3525
 - getPriority, 3525
 - getState, 3525
 - getUncaughtExceptionHandler, 3525
 - isAlive, 3526
 - join, 3526
 - MAX_PRIORITY, 3529
 - MIN_PRIORITY, 3529
 - NEW, 3523
 - NORM_PRIORITY, 3529
 - run, 3527
 - RUNNABLE, 3523
 - setName, 3527
 - setPriority, 3527
 - setUncaughtExceptionHandler, 3527
 - sleep, 3528
 - SLEEPING, 3524
 - start, 3528
 - State, 3523
 - TERMINATED, 3524
 - Thread, 3524
 - TIMED_WAITING, 3524
 - toString, 3529
 - WAITING, 3523
 - yield, 3529
- decaf::lang::Thread::UncaughtExceptionHandler, 3648
 - ~UncaughtExceptionHandler, 3649
 - uncaughtException, 3649
- decaf::lang::ThreadGroup, 3531
 - ~ThreadGroup, 3531
 - ThreadGroup, 3531
- decaf::lang::Throwable, 3536
 - ~Throwable, 3538
 - clone, 3538
 - getCause, 3539
 - getMessage, 3539
 - getStackTrace, 3539
 - getStackTraceString, 3539
 - initCause, 3540
 - printStackTrace, 3540
 - setMark, 3540
 - Throwable, 3538
- decaf::net, 130
- decaf::net::BindException, 768
 - ~BindException, 770
 - BindException, 769, 770
 - clone, 770
- decaf::net::ConnectException, 1165
 - ~ConnectException, 1167
 - clone, 1167
 - ConnectException, 1166, 1167
- decaf::net::HttpRetryException, 1858
 - ~HttpRetryException, 1860
 - clone, 1861
 - HttpRetryException, 1859, 1860
- decaf::net::Inet4Address, 1880
 - ~Inet4Address, 1881
 - Inet4Address, 1881
 - InetAddress, 1883
 - isAnyLocalAddress, 1881
 - isLinkLocalAddress, 1881
 - isLoopbackAddress, 1881
 - isMCGlobal, 1882
 - isMCLinkLocal, 1882
 - isMCNodeLocal, 1882
 - isMCOrgLocal, 1882
 - isMCSiteLocal, 1882
 - isMulticastAddress, 1883
 - isSiteLocalAddress, 1883
- decaf::net::Inet6Address, 1883
 - ~Inet6Address, 1884
 - Inet6Address, 1884
 - InetAddress, 1884
- decaf::net::InetAddress, 1884
 - ~InetAddress, 1887
 - addressBytes, 1891
 - ANY, 1891
 - anyBytes, 1891
 - bytesToInt, 1887
 - getAddress, 1887
 - getByAddress, 1887, 1888
 - getHostAddress, 1888
 - getHostName, 1888
 - getLocalHost, 1888
 - hostname, 1891
 - InetAddress, 1887
 - isAnyLocalAddress, 1888
 - isLinkLocalAddress, 1889
 - isLoopbackAddress, 1889
 - isMCGlobal, 1889
 - isMCLinkLocal, 1889
 - isMCNodeLocal, 1889

- isMCOrgLocal, 1890
- isMCSiteLocal, 1890
- isMulticastAddress, 1890
- isSiteLocalAddress, 1890
- LOOPBACK, 1891
- loopbackBytes, 1891
- reached, 1891
- toString, 1890
- decaf::net::InetSocketAddress, 1891
 - ~InetSocketAddress, 1892
 - InetSocketAddress, 1892
- decaf::net::MalformedURLException, 2303
 - ~MalformedURLException, 2305
 - clone, 2305
 - MalformedURLException, 2303, 2304
- decaf::net::NoRouteToHostException, 2640
 - ~NoRouteToHostException, 2642
 - clone, 2642
 - NoRouteToHostException, 2641, 2642
- decaf::net::PortUnreachableException, 2781
 - ~PortUnreachableException, 2783
 - clone, 2783
 - PortUnreachableException, 2781, 2782
- decaf::net::ProtocolException, 2937
 - ~ProtocolException, 2939
 - clone, 2939
 - ProtocolException, 2938, 2939
- decaf::net::ServerSocket, 3136
 - ~ServerSocket, 3140
 - accept, 3141
 - bind, 3141
 - checkClosed, 3142
 - close, 3142
 - ensureCreated, 3142
 - getDefaultBacklog, 3142
 - getLocalPort, 3142
 - getReceiveBufferSize, 3142
 - getReuseAddress, 3143
 - getSoTimeout, 3143
 - implAccept, 3143
 - isBound, 3143
 - isClosed, 3143
 - ServerSocket, 3139, 3140
 - setReceiveBufferSize, 3144
 - setReuseAddress, 3144
 - setSocketImplFactory, 3144
 - setSoTimeout, 3144
 - setupSocketImpl, 3145
 - toString, 3145
- decaf::net::ServerSocketFactory, 3145
 - ~ServerSocketFactory, 3146
 - createServerSocket, 3146, 3147
 - getDefault, 3148
 - ServerSocketFactory, 3146
- decaf::net::Socket, 3281
 - ~Socket, 3287
 - accepted, 3287
 - bind, 3287
 - checkClosed, 3287
 - close, 3287
 - connect, 3287, 3288
 - ensureCreated, 3288
 - getInetAddress, 3288
 - getInputStream, 3288
 - getKeepAlive, 3289
 - getLocalAddress, 3289
 - getLocalPort, 3289
 - getOOBInline, 3289
 - getOutputStream, 3290
 - getPort, 3290
 - getReceiveBufferSize, 3290
 - getReuseAddress, 3290
 - getSendBufferSize, 3291
 - getSoLinger, 3291
 - getSoTimeout, 3291
 - getTcpNoDelay, 3291
 - getTrafficClass, 3292
 - impl, 3297
 - initSocketImpl, 3292
 - isBound, 3292
 - isClosed, 3292
 - isConnected, 3292
 - isInputShutdown, 3293
 - isOutputShutdown, 3293
 - sendUrgentData, 3293
 - ServerSocket, 3297
 - setKeepAlive, 3293
 - setOOBInline, 3293
 - setReceiveBufferSize, 3294
 - setReuseAddress, 3294
 - setSendBufferSize, 3294
 - setSocketImplFactory, 3295
 - setSoLinger, 3295
 - setSoTimeout, 3295
 - setTcpNoDelay, 3296
 - setTrafficClass, 3296
 - shutdownInput, 3296
 - shutdownOutput, 3296
 - Socket, 3285, 3286
 - toString, 3297
- decaf::net::SocketAddress, 3297
 - ~SocketAddress, 3298
- decaf::net::SocketError, 3298
 - getErrorCode, 3298
 - getErrorString, 3298
- decaf::net::SocketException, 3298
 - ~SocketException, 3300
 - clone, 3300

- SocketException, 3299, 3300
- decaf::net::SocketFactory, 3301
 - ~SocketFactory, 3302
 - createSocket, 3302–3304
 - getDefault, 3304
 - SocketFactory, 3302
- decaf::net::SocketImpl, 3306
 - ~SocketImpl, 3308
 - accept, 3308
 - address, 3313
 - available, 3308
 - bind, 3308
 - close, 3309
 - connect, 3309
 - create, 3309
 - fd, 3313
 - getFileDescriptor, 3309
 - getInetAddress, 3310
 - getInputStream, 3310
 - getLocalAddress, 3310
 - getLocalPort, 3310
 - getOption, 3310
 - getOutputStream, 3311
 - getPort, 3311
 - listen, 3311
 - localPort, 3313
 - port, 3313
 - sendUrgentData, 3312
 - setOption, 3312
 - shutdownInput, 3312
 - shutdownOutput, 3312
 - SocketImpl, 3308
 - supportsUrgentData, 3313
 - toString, 3313
- decaf::net::SocketImplFactory, 3314
 - ~SocketImplFactory, 3314
 - createSocketImpl, 3314
- decaf::net::SocketOptions, 3315
 - ~SocketOptions, 3316
 - SOCKET_OPTION_BINDADDR, 3316
 - SOCKET_OPTION_BROADCAST, 3316
 - SOCKET_OPTION_IP_-MULTICAST_IF, 3316
 - SOCKET_OPTION_IP_-MULTICAST_IF2, 3316
 - SOCKET_OPTION_IP_-MULTICAST_LOOP, 3317
 - SOCKET_OPTION_IP_TOS, 3317
 - SOCKET_OPTION_KEEPAIVE, 3317
 - SOCKET_OPTION_LINGER, 3317
 - SOCKET_OPTION_OOBLINE, 3317
 - SOCKET_OPTION_RCVBUF, 3318
 - SOCKET_OPTION_REUSEADDR, 3318
 - SOCKET_OPTION_SNDBUF, 3318
 - SOCKET_OPTION_TCP_NODELAY, 3318
 - SOCKET_OPTION_TIMEOUT, 3318
- decaf::net::SocketTimeoutException, 3319
 - ~SocketTimeoutException, 3320
 - clone, 3321
 - SocketTimeoutException, 3319, 3320
- decaf::net::ssl, 132
- decaf::net::ssl::SSLContext, 3321
 - ~SSLContext, 3322
 - getDefault, 3322
 - getDefaultSSLParameters, 3322
 - getServerSocketFactory, 3322
 - getSocketFactory, 3323
 - getSupportedSSLParameters, 3323
 - setDefault, 3323
 - SSLContext, 3322
- decaf::net::ssl::SSLContextSpi, 3324
 - ~SSLContextSpi, 3324
 - providerGetDefaultSSLParameters, 3324
 - providerGetServerSocketFactory, 3325
 - providerGetSocketFactory, 3325
 - providerGetSupportedSSLParameters, 3325
 - providerInit, 3326
- decaf::net::ssl::SSLParameters, 3326
 - ~SSLParameters, 3328
 - getCipherSuites, 3328
 - getNeedClientAuth, 3328
 - getProtocols, 3328
 - getWantClientAuth, 3328
 - setCipherSuites, 3328
 - setNeedClientAuth, 3328
 - setProtocols, 3329
 - setWantClientAuth, 3329
 - SSLParameters, 3327
- decaf::net::ssl::SSLServerSocket, 3329
 - ~SSLServerSocket, 3332
 - getEnabledCipherSuites, 3332
 - getEnabledProtocols, 3332
 - getNeedClientAuth, 3333
 - getSupportedCipherSuites, 3333
 - getSupportedProtocols, 3333
 - getWantClientAuth, 3333
 - setEnabledCipherSuites, 3333
 - setEnabledProtocols, 3334
 - setNeedClientAuth, 3334
 - setWantClientAuth, 3334
 - SSLServerSocket, 3331
- decaf::net::ssl::SSLServerSocketFactory, 3335
 - ~SSLServerSocketFactory, 3336

- getDefault, 3336
- getDefaultCipherSuites, 3336
- getSupportedCipherSuites, 3336
- SSLServerSocketFactory, 3336
- decaf::net::ssl::SSLSocket, 3337
 - ~SSLSocket, 3340
 - getEnabledCipherSuites, 3340
 - getEnabledProtocols, 3340
 - getNeedClientAuth, 3341
 - getSSLParameters, 3341
 - getSupportedCipherSuites, 3341
 - getSupportedProtocols, 3341
 - getUseClientMode, 3342
 - getWantClientAuth, 3342
 - setEnabledCipherSuites, 3342
 - setEnabledProtocols, 3342
 - setNeedClientAuth, 3343
 - setSSLParameters, 3343
 - setUseClientMode, 3343
 - setWantClientAuth, 3344
 - SSLSocket, 3339, 3340
 - startHandshake, 3344
- decaf::net::ssl::SSLSocketFactory, 3345
 - ~SSLSocketFactory, 3346
 - createSocket, 3346
 - getDefault, 3346
 - getDefaultCipherSuites, 3346
 - getSupportedCipherSuites, 3347
 - SSLSocketFactory, 3346
- decaf::net::UnknownHostException, 3649
 - ~UnknownHostException, 3651
 - clone, 3651
 - UnknownHostException, 3650, 3651
- decaf::net::UnknownServiceException, 3652
 - ~UnknownServiceException, 3653
 - clone, 3654
 - UnknownServiceException, 3652, 3653
- decaf::net::URL, 3660
 - ~URL, 3664
 - compareTo, 3665
 - create, 3665
 - equals, 3665
 - getAuthority, 3665
 - getFragment, 3665
 - getHost, 3665
 - getPath, 3666
 - getPort, 3666
 - getQuery, 3666
 - getRawAuthority, 3666
 - getRawFragment, 3666
 - getRawPath, 3666
 - getRawQuery, 3667
 - getRawSchemeSpecificPart, 3667
 - getRawUserInfo, 3667
 - getScheme, 3667
 - getSchemeSpecificPart, 3667
 - getUserInfo, 3668
 - isAbsolute, 3668
 - isOpaque, 3668
 - normalize, 3668
 - operator<, 3669
 - operator==, 3669
 - parseServerAuthority, 3669
 - relativize, 3670
 - resolve, 3670
 - toString, 3671
 - toURL, 3671
 - URI, 3663, 3664
- decaf::net::URISyntaxException, 3686
 - ~URISyntaxException, 3689
 - clone, 3689
 - getIndex, 3689
 - getInput, 3690
 - getReason, 3690
 - URISyntaxException, 3687–3689
- decaf::net::URL, 3697
 - ~URL, 3699
 - URL, 3699
- decaf::net::URLDecoder, 3699
 - ~URLDecoder, 3700
 - decode, 3700
- decaf::net::URLEncoder, 3700
 - ~URLEncoder, 3701
 - encode, 3701
- decaf::nio, 132
- decaf::nio::Buffer, 855
 - ~Buffer, 858
 - _capacity, 861
 - _limit, 861
 - _mark, 861
 - _markSet, 861
 - _position, 861
 - Buffer, 858
 - capacity, 858
 - clear, 858
 - flip, 858
 - hasRemaining, 859
 - isReadOnly, 859
 - limit, 859
 - mark, 860
 - position, 860
 - remaining, 860
 - reset, 860
 - rewind, 861
- decaf::nio::BufferOverflowException, 880
 - ~BufferOverflowException, 882
 - BufferOverflowException, 880, 881
 - clone, 882

- decaf::nio::BufferUnderflowException, 882
 - ~BufferUnderflowException, 884
 - BufferUnderflowException, 883, 884
 - clone, 884
- decaf::nio::ByteBuffer, 954
 - ~ByteBuffer, 959
 - allocate, 959
 - array, 960
 - arrayOffset, 960
 - asCharBuffer, 960
 - asDoubleBuffer, 961
 - asFloatBuffer, 961
 - asIntBuffer, 961
 - asLongBuffer, 962
 - asReadOnlyBuffer, 962
 - asShortBuffer, 962
 - ByteBuffer, 959
 - compact, 963
 - compareTo, 963
 - duplicate, 963
 - equals, 964
 - get, 964, 965
 - getChar, 965, 966
 - getDouble, 966
 - getFloat, 967
 - getInt, 967, 968
 - getLong, 968
 - getShort, 969
 - hasArray, 969
 - isReadOnly, 970
 - operator<, 970
 - operator==, 970
 - put, 970–972
 - putChar, 972, 973
 - putDouble, 973, 974
 - putFloat, 974
 - putInt, 975
 - putLong, 976
 - putShort, 976, 977
 - slice, 977
 - toString, 978
 - wrap, 978
- decaf::nio::CharBuffer, 1037
 - ~CharBuffer, 1041
 - allocate, 1041
 - append, 1041, 1042
 - array, 1042
 - arrayOffset, 1043
 - asReadOnlyBuffer, 1043
 - charAt, 1044
 - CharBuffer, 1041
 - compact, 1044
 - compareTo, 1044
 - duplicate, 1044
 - equals, 1045
 - get, 1045, 1046
 - hasArray, 1046
 - length, 1047
 - operator<, 1047
 - operator==, 1047
 - put, 1047–1050
 - read, 1050
 - slice, 1051
 - subSequence, 1051
 - toString, 1052
 - wrap, 1052
- decaf::nio::DoubleBuffer, 1692
 - ~DoubleBuffer, 1695
 - allocate, 1695
 - array, 1696
 - arrayOffset, 1696
 - asReadOnlyBuffer, 1696
 - compact, 1697
 - compareTo, 1697
 - DoubleBuffer, 1695
 - duplicate, 1697
 - equals, 1697
 - get, 1698, 1699
 - hasArray, 1699
 - operator<, 1699
 - operator==, 1700
 - put, 1700, 1701
 - slice, 1702
 - toString, 1702
 - wrap, 1702, 1703
- decaf::nio::FloatBuffer, 1800
 - ~FloatBuffer, 1803
 - allocate, 1803
 - array, 1803
 - arrayOffset, 1803
 - asReadOnlyBuffer, 1804
 - compact, 1804
 - compareTo, 1805
 - duplicate, 1805
 - equals, 1805
 - FloatBuffer, 1802
 - get, 1805, 1806
 - hasArray, 1807
 - operator<, 1807
 - operator==, 1807
 - put, 1807–1809
 - slice, 1809
 - toString, 1810
 - wrap, 1810
- decaf::nio::IntBuffer, 1931
 - ~IntBuffer, 1933
 - allocate, 1933
 - array, 1934

- arrayOffset, 1934
- asReadOnlyBuffer, 1934
- compact, 1935
- compareTo, 1935
- duplicate, 1935
- equals, 1935
- get, 1936, 1937
- hasArray, 1937
- IntBuffer, 1933
- operator<, 1937
- operator==, 1938
- put, 1938, 1939
- slice, 1940
- toString, 1940
- wrap, 1940, 1941
- decaf::nio::InvalidMarkException, 1997
 - ~InvalidMarkException, 1999
 - clone, 1999
 - InvalidMarkException, 1997, 1998
- decaf::nio::LongBuffer, 2291
 - ~LongBuffer, 2294
 - allocate, 2294
 - array, 2294
 - arrayOffset, 2294
 - asReadOnlyBuffer, 2295
 - compact, 2295
 - compareTo, 2295
 - duplicate, 2296
 - equals, 2296
 - get, 2296, 2297
 - hasArray, 2297
 - LongBuffer, 2293
 - operator<, 2298
 - operator==, 2298
 - put, 2298–2300
 - slice, 2300
 - toString, 2301
 - wrap, 2301
- decaf::nio::ReadOnlyBufferException, 2966
 - ~ReadOnlyBufferException, 2968
 - clone, 2968
 - ReadOnlyBufferException, 2967, 2968
- decaf::nio::ShortBuffer, 3238
 - ~ShortBuffer, 3241
 - allocate, 3241
 - array, 3241
 - arrayOffset, 3242
 - asReadOnlyBuffer, 3242
 - compact, 3243
 - compareTo, 3243
 - duplicate, 3243
 - equals, 3243
 - get, 3244, 3245
 - hasArray, 3245
 - operator<, 3245
 - operator==, 3246
 - put, 3246, 3247
 - ShortBuffer, 3241
 - slice, 3248
 - toString, 3248
 - wrap, 3248, 3249
- decaf::security, 133
- decaf::security::auth, 133
- decaf::security::auth::x500, 133
- decaf::security::auth::x500::X500Principal, 3761
 - ~X500Principal, 3762
 - getEncoded, 3762
 - getName, 3762
 - hashCode, 3762
- decaf::security::cert, 134
- decaf::security::cert::Certificate, 1007
 - ~Certificate, 1008
 - equals, 1008
 - getEncoded, 1008
 - getPublicKey, 1008
 - getType, 1008
 - toString, 1009
 - verify, 1009
- decaf::security::cert::CertificateEncodingException, 1010
 - ~CertificateEncodingException, 1011
 - CertificateEncodingException, 1010, 1011
 - clone, 1011
- decaf::security::cert::CertificateException, 1012
 - ~CertificateException, 1013
 - CertificateException, 1012, 1013
 - clone, 1013
- decaf::security::cert::CertificateExpiredException, 1014
 - ~CertificateExpiredException, 1015
 - CertificateExpiredException, 1014, 1015
 - clone, 1015
- decaf::security::cert::CertificateNotYetValidException, 1015
 - ~CertificateNotYetValidException, 1017
 - CertificateNotYetValidException, 1016
 - clone, 1017
- decaf::security::cert::CertificateParsingException, 1017
 - ~CertificateParsingException, 1019
 - CertificateParsingException, 1018
 - clone, 1019
- decaf::security::cert::X509Certificate, 3762
 - ~X509Certificate, 3764
 - checkValidity, 3764
 - getBasicConstraints, 3764
 - getIssuerUniqueID, 3764
 - getIssuerX500Principal, 3764

- getKeyUsage, 3764
- getNotAfter, 3764
- getNotBefore, 3764
- getSigAlgName, 3764
- getSigAlgOID, 3764
- getSigAlgParams, 3764
- getSignature, 3764
- getSubjectUniqueID, 3764
- getSubjectX500Principal, 3764
- getTBSCertificate, 3764
- getVersion, 3764
- decaf::security::GeneralSecurityException, 1845
 - ~GeneralSecurityException, 1847
 - clone, 1847
 - GeneralSecurityException, 1845, 1846
- decaf::security::InvalidKeyException, 1994
 - ~InvalidKeyException, 1996
 - clone, 1996
 - InvalidKeyException, 1995, 1996
- decaf::security::Key, 2149
 - ~Key, 2150
 - getAlgorithm, 2150
 - getEncoded, 2150
 - getFormat, 2150
- decaf::security::KeyException, 2151
 - ~KeyException, 2153
 - clone, 2153
 - KeyException, 2151, 2152
- decaf::security::KeyManagementException, 2153
 - ~KeyManagementException, 2155
 - clone, 2155
 - KeyManagementException, 2154, 2155
- decaf::security::NoSuchAlgorithmException, 2643
 - ~NoSuchAlgorithmException, 2645
 - clone, 2645
 - NoSuchAlgorithmException, 2644
- decaf::security::NoSuchProviderException, 2648
 - ~NoSuchProviderException, 2650
 - clone, 2650
 - NoSuchProviderException, 2649
- decaf::security::Principal, 2831
 - ~Principal, 2831
 - equals, 2831
 - getName, 2831
- decaf::security::PublicKey, 2940
 - ~PublicKey, 2940
- decaf::security::SecureRandom, 3116
 - ~SecureRandom, 3119
 - next, 3119
 - nextBytes, 3119, 3120
 - SecureRandom, 3118
 - setSeed, 3120
- decaf::security::SecureRandomSpi, 3124
 - ~SecureRandomSpi, 3125
 - providerGenerateSeed, 3125
 - providerNextBytes, 3125
 - providerSetSeed, 3126
 - SecureRandomSpi, 3125
- decaf::security::SignatureException, 3276
 - ~SignatureException, 3278
 - clone, 3278
 - SignatureException, 3276, 3277
- decaf::util, 134
- decaf::util::AbstractCollection, 143
 - ~AbstractCollection, 146
 - AbstractCollection, 146
 - add, 146
 - addAll, 146
 - clear, 147
 - contains, 148
 - containsAll, 148
 - copy, 149
 - equals, 149
 - isEmpty, 149
 - lock, 150
 - mutex, 155
 - notify, 150
 - notifyAll, 150
 - operator=, 151
 - remove, 151
 - removeAll, 152
 - retainAll, 153
 - toArray, 153
 - tryLock, 153
 - unlock, 154
 - wait, 154, 155
- decaf::util::AbstractList, 156
 - ~AbstractList, 157
- decaf::util::AbstractMap, 157
 - ~AbstractMap, 158
- decaf::util::AbstractQueue, 158
 - ~AbstractQueue, 159
 - AbstractQueue, 159
 - add, 159
 - addAll, 159
 - clear, 160
 - element, 160
 - remove, 161
- decaf::util::AbstractSequentialList, 161
 - ~AbstractSequentialList, 162
- decaf::util::AbstractSet, 162
 - ~AbstractSet, 163
 - removeAll, 163
- decaf::util::Collection, 1097
 - ~Collection, 1099
 - add, 1099

- addAll, 1100
- clear, 1100
- contains, 1101
- containsAll, 1102
- equals, 1102
- isEmpty, 1103
- remove, 1104
- removeAll, 1105
- retainAll, 1105
- size, 1106
- toArray, 1107
- decaf::util::Comparator, 1127
 - ~Comparator, 1128
- compare, 1128
- operator(), 1129
- decaf::util::comparators, 136
- decaf::util::comparators::Less, 2182
 - ~Less, 2183
- compare, 2183
- Less, 2183
- operator(), 2183
- decaf::util::concurrent, 136
- decaf::util::concurrent::atomic, 138
- decaf::util::concurrent::atomic::AtomicBoolean, 677
 - ~AtomicBoolean, 678
- AtomicBoolean, 678
- compareAndSet, 678
- get, 679
- getAndSet, 679
- set, 679
- toString, 679
- decaf::util::concurrent::atomic::AtomicInteger, 680
 - ~AtomicInteger, 681
- addAndGet, 681
- AtomicInteger, 681
- compareAndSet, 682
- decrementAndGet, 682
- doubleValue, 682
- floatValue, 682
- get, 683
- getAndAdd, 683
- getAndDecrement, 683
- getAndIncrement, 683
- getAndSet, 683
- incrementAndGet, 684
- intValue, 684
- longValue, 684
- set, 684
- toString, 684
- decaf::util::concurrent::atomic::AtomicRefCounter, 685
 - ~AtomicRefCounter, 686
- AtomicRefCounter, 686
- release, 686
- swap, 687
- decaf::util::concurrent::atomic::AtomicReference, 687
 - ~AtomicReference, 688
- AtomicReference, 688
- compareAndSet, 688
- get, 689
- getAndSet, 689
- set, 689
- toString, 689
- decaf::util::concurrent::BlockingQueue, 774
 - ~BlockingQueue, 777
- drainTo, 777, 778
- offer, 778
- poll, 779
- put, 779
- remainingCapacity, 780
- take, 780
- decaf::util::concurrent::BrokenBarrierException, 790
 - ~BrokenBarrierException, 792
- BrokenBarrierException, 791, 792
- clone, 792
- decaf::util::concurrent::Callable, 1003
 - ~Callable, 1004
- call, 1004
- decaf::util::concurrent::CancellationException, 1004
 - ~CancellationException, 1006
- CancellationException, 1005, 1006
- clone, 1006
- decaf::util::concurrent::ConcurrentMap, 1136
 - ~ConcurrentMap, 1137
- putIfAbsent, 1137
- remove, 1138
- replace, 1139
- decaf::util::concurrent::ConcurrentStlMap, 1140
 - ~ConcurrentStlMap, 1145
- clear, 1145
- ConcurrentStlMap, 1145
- containsKey, 1145
- containsValue, 1146
- copy, 1146
- equals, 1147
- get, 1147, 1148
- isEmpty, 1148
- keySet, 1148
- lock, 1149
- notify, 1149
- notifyAll, 1149
- put, 1149
- putAll, 1150

- putIfAbsent, 1150
- remove, 1151, 1152
- replace, 1152, 1153
- size, 1153
- tryLock, 1153
- unlock, 1154
- values, 1154
- wait, 1154, 1155
- decaf::util::concurrent::ConditionHandle, 1162
 - ~ConditionHandle, 1163
 - condition, 1163
 - ConditionHandle, 1163
 - criticalSection, 1163
 - generation, 1163
 - mutex, 1163
 - numWaiting, 1163
 - numWake, 1163
 - semaphore, 1163
- decaf::util::concurrent::CountDownLatch, 1417
 - ~CountDownLatch, 1418
 - await, 1418, 1419
 - countDown, 1419
 - CountDownLatch, 1417
 - getCount, 1419
- decaf::util::concurrent::Delayed, 1608
 - ~Delayed, 1609
 - getDelay, 1609
- decaf::util::concurrent::ExecutionException, 1746
 - ~ExecutionException, 1748
 - clone, 1749
 - ExecutionException, 1747, 1748
- decaf::util::concurrent::Executor, 1749
 - ~Executor, 1750
 - execute, 1750
- decaf::util::concurrent::ExecutorService, 1751
 - ~ExecutorService, 1752
 - awaitTermination, 1752
- decaf::util::concurrent::Future, 1840
 - ~Future, 1841
 - cancel, 1841
 - get, 1842
 - isCancelled, 1842
 - isDone, 1843
- decaf::util::concurrent::Lock, 2228
 - ~Lock, 2229
 - isLocked, 2229
 - Lock, 2228
 - lock, 2229
 - unlock, 2229
- decaf::util::concurrent::locks, 138
- decaf::util::concurrent::locks::Condition, 1156
 - ~Condition, 1158
 - await, 1158, 1159
 - awaitNanos, 1159
 - awaitUninterruptibly, 1161
 - awaitUntil, 1161
 - signal, 1162
 - signalAll, 1162
- decaf::util::concurrent::locks::Lock, 2229
 - ~Lock, 2231
 - lock, 2231
 - lockInterruptibly, 2231
 - newCondition, 2232
 - tryLock, 2232, 2233
 - unlock, 2234
- decaf::util::concurrent::locks::LockSupport, 2234
 - ~LockSupport, 2236
 - decaf::lang::Thread, 3529
 - park, 2236
 - parkNanos, 2236
 - parkUntil, 2237
 - unpark, 2237
- decaf::util::concurrent::locks::ReadWriteLock, 2968
 - ~ReadWriteLock, 2970
 - readLock, 2970
 - writeLock, 2970
- decaf::util::concurrent::locks::ReentrantLock, 2977
 - ~ReentrantLock, 2979
 - getHoldCount, 2979
 - isFair, 2980
 - isHeldByCurrentThread, 2980
 - isLocked, 2980
 - lock, 2980
 - lockInterruptibly, 2981
 - newCondition, 2981
 - ReentrantLock, 2979
 - toString, 2982
 - tryLock, 2982, 2983
 - unlock, 2984
- decaf::util::concurrent::Mutex, 2604
 - ~Mutex, 2605
 - lock, 2605
 - Mutex, 2605
 - notify, 2605
 - notifyAll, 2606
 - tryLock, 2606
 - unlock, 2607
 - wait, 2607, 2608
- decaf::util::concurrent::MutexHandle, 2609
 - ~MutexHandle, 2609
 - lock_count, 2609
 - lock_owner, 2609
 - mutex, 2609
 - MutexHandle, 2609

- decaf::util::concurrent::PooledThread, 2777
 - ~PooledThread, 2778
 - getPooledThreadListener, 2778
 - isBusy, 2778
 - PooledThread, 2778
 - run, 2778
 - setPooledThreadListener, 2778
 - stop, 2779
- decaf::util::concurrent::PooledThreadListener, 2779
 - ~PooledThreadListener, 2780
 - onTaskCompleted, 2780
 - onTaskException, 2780
 - onTaskStarted, 2780
- decaf::util::concurrent::RejectedExecutionException, 2984
 - ~RejectedExecutionException, 2986
 - clone, 2986
 - RejectedExecutionException, 2985, 2986
- decaf::util::concurrent::RejectedExecutionHandler, 2987
 - ~RejectedExecutionHandler, 2987
 - rejectedExecution, 2987
- decaf::util::concurrent::Semaphore, 3126
 - ~Semaphore, 3129
 - acquire, 3129, 3130
 - acquireUninterruptibly, 3130, 3131
 - availablePermits, 3131
 - drainPermits, 3131
 - isFair, 3131
 - release, 3132
 - Semaphore, 3129
 - toString, 3132
 - tryAcquire, 3132–3134
- decaf::util::concurrent::Synchronizable, 3461
 - ~Synchronizable, 3463
 - lock, 3463
 - notify, 3464
 - notifyAll, 3465
 - tryLock, 3466
 - unlock, 3467
 - wait, 3468, 3470, 3471
- decaf::util::concurrent::SynchronousQueue, 3477
 - ~SynchronousQueue, 3480
 - clear, 3480
 - contains, 3480
 - containsAll, 3480
 - drainTo, 3481
 - equals, 3482
 - isEmpty, 3482
 - iterator, 3482, 3483
 - offer, 3483
 - peek, 3484
 - poll, 3484
 - put, 3484
 - remainingCapacity, 3485
 - remove, 3485
 - removeAll, 3486
 - retainAll, 3486
 - size, 3486
 - SynchronousQueue, 3480
 - take, 3486
 - toArray, 3486
- decaf::util::concurrent::TaskListener, 3495
 - ~TaskListener, 3496
 - onTaskComplete, 3496
 - onTaskException, 3496
- decaf::util::concurrent::ThreadFactory, 3529
 - ~ThreadFactory, 3530
 - newThread, 3530
- decaf::util::concurrent::ThreadPool, 3531
 - ~ThreadPool, 3533
 - DEFAULT_MAX_BLOCK_SIZE, 3536
 - DEFAULT_MAX_POOL_SIZE, 3536
 - deQueueTask, 3533
 - getBacklog, 3533
 - getBlockSize, 3533
 - getFreeThreadCount, 3534
 - getInstance, 3534
 - getMaxThreads, 3534
 - getPoolSize, 3534
 - onTaskCompleted, 3534
 - onTaskException, 3535
 - onTaskStarted, 3535
 - queueTask, 3535
 - reserve, 3535
 - setBlockSize, 3536
 - setMaxThreads, 3536
 - Task, 3533
 - ThreadPool, 3533
- decaf::util::concurrent::TimeoutException, 3541
 - ~TimeoutException, 3542
 - clone, 3543
 - TimeoutException, 3541, 3542
- decaf::util::concurrent::TimeUnit, 3559
 - ~TimeUnit, 3561
 - compareTo, 3561
 - convert, 3562
 - DAYS, 3567
 - equals, 3562
 - HOURS, 3567
 - MICROSECONDS, 3567
 - MILLISECONDS, 3567
 - MINUTES, 3567
 - NANOSECONDS, 3567
 - operator<, 3562
 - operator==, 3563

- SECONDS, 3567
- sleep, 3563
- timedJoin, 3563
- timedWait, 3564
- TimeUnit, 3561
- toDays, 3564
- toHours, 3565
- toMicros, 3565
- toMillis, 3565
- toMinutes, 3566
- toNanos, 3566
- toSeconds, 3566
- toString, 3567
- valueOf, 3567
- values, 3568
- decaf::util::Date, 1559
 - ~Date, 1560
 - after, 1560
 - before, 1560
 - compareTo, 1561
 - Date, 1560
 - equals, 1561
 - getTime, 1561
 - operator<, 1561
 - operator=, 1562
 - operator==, 1562
 - setTime, 1562
 - toString, 1562
- decaf::util::Iterator, 2012
 - ~Iterator, 2013
 - hasNext, 2013
 - next, 2013
 - remove, 2013
- decaf::util::List, 2190
 - ~List, 2192
 - add, 2192
 - addAll, 2192
 - get, 2193
 - indexOf, 2193
 - lastIndexOf, 2194
 - List, 2192
 - listIterator, 2194, 2195
 - remove, 2196
 - set, 2196
- decaf::util::ListIterator, 2197
 - ~ListIterator, 2198
 - add, 2198
 - hasPrevious, 2198
 - nextIndex, 2199
 - previous, 2199
 - previousIndex, 2199
 - set, 2199
- decaf::util::logging, 139
 - Debug, 140
 - Error, 140
 - Fatal, 140
 - Info, 140
 - Levels, 140
 - Markblock, 140
 - Null, 140
 - Off, 140
 - Throwing, 140
 - Warn, 140
- decaf::util::logging::ConsoleHandler, 1300
 - ~ConsoleHandler, 1301
 - close, 1301
 - ConsoleHandler, 1301
 - publish, 1301
- decaf::util::logging::ErrorManager, 1710
 - ~ErrorManager, 1711
 - CLOSE_FAILURE, 1711
 - error, 1711
 - ErrorManager, 1711
 - FLUSH_FAILURE, 1711
 - FORMAT_FAILURE, 1711
 - GENERIC_FAILURE, 1711
 - OPEN_FAILURE, 1711
 - WRITE_FAILURE, 1711
- decaf::util::logging::Filter, 1770
 - ~Filter, 1770
 - isLoggable, 1770
- decaf::util::logging::Formatter, 1838
 - ~Formatter, 1839
 - format, 1839
 - formatMessage, 1839
 - getHead, 1840
 - getTail, 1840
- decaf::util::logging::Handler, 1852
 - ~Handler, 1853
 - flush, 1853
 - getErrorManager, 1853
 - getFilter, 1853
 - getFormatter, 1853
 - getLevel, 1854
 - Handler, 1853
 - isLoggable, 1854
 - publish, 1854
 - reportError, 1854
 - setErrorManager, 1855
 - setFilter, 1855
 - setFormatter, 1855
 - setLevel, 1855
- decaf::util::logging::Level, 2185
 - ~Level, 2187
 - ALL, 2189
 - compareTo, 2188
 - CONFIG, 2189
 - DEBUG, 2189

- equality, 2188
- FINE, 2189
- FINER, 2189
- FINEST, 2189
- getName, 2188
- INFO, 2190
- INHERIT, 2190
- intValue, 2188
- Level, 2187
- OFF, 2190
- operator<, 2188
- operator==, 2188
- parse, 2188
- SEVERE, 2190
- toString, 2188
- WARNING, 2190
- decaf::util::logging::Logger, 2237
 - ~Logger, 2241
 - addHandler, 2241
 - config, 2241
 - debug, 2241
 - entering, 2242
 - exiting, 2242
 - fine, 2242
 - finer, 2243
 - finest, 2243
 - getAnonymousLogger, 2243
 - getFilter, 2243
 - getHandlers, 2244
 - getLevel, 2244
 - getLogger, 2244
 - getName, 2244
 - getParent, 2245
 - getUseParentHandlers, 2245
 - info, 2245
 - isLoggable, 2245
 - log, 2246
 - Logger, 2241
 - removeHandler, 2247
 - setFilter, 2247
 - setLevel, 2247
 - setParent, 2248
 - setUseParentHandlers, 2248
 - severe, 2248
 - throwing, 2248
 - warning, 2249
- decaf::util::logging::LoggerHierarchy, 2249
 - ~LoggerHierarchy, 2249
 - LoggerHierarchy, 2249
- decaf::util::logging::LogManager, 2254
 - ~LogManager, 2257
 - addLogger, 2258
 - addPropertyChangeListener, 2258
 - decaf::lang::Runtime, 2261
 - getLogger, 2258
 - getLoggerNames, 2258
 - getLogManager, 2258
 - getProperties, 2259
 - getProperty, 2259
 - LogManager, 2257
 - operator=, 2259
 - readConfiguration, 2259, 2260
 - removePropertyChangeListener, 2260
 - reset, 2260
 - setProperties, 2260
- decaf::util::logging::LogRecord, 2261
 - ~LogRecord, 2262
 - getLevel, 2262
 - getLoggerName, 2262
 - getMessage, 2263
 - getSourceFile, 2263
 - getSourceFunction, 2263
 - getSourceLine, 2263
 - getThrown, 2263
 - getTimestamp, 2263
 - getThreadId, 2264
 - LogRecord, 2262
 - setLevel, 2264
 - setLoggerName, 2264
 - setMessage, 2264
 - setSourceFile, 2264
 - setSourceFunction, 2264
 - setSourceLine, 2265
 - setThrown, 2265
 - setTimestamp, 2265
 - setThreadId, 2265
- decaf::util::logging::LogWriter, 2266
 - ~LogWriter, 2266
 - destroy, 2266
 - getInstance, 2266
 - log, 2266, 2267
 - LogWriter, 2266
 - returnInstance, 2267
- decaf::util::logging::MarkBlockLogger, 2327
 - ~MarkBlockLogger, 2328
 - MarkBlockLogger, 2328
- decaf::util::logging::PropertiesChangeListener, 2936
 - ~PropertiesChangeListener, 2937
 - onPropertiesReset, 2937
 - onPropertyChanged, 2937
- decaf::util::logging::SimpleFormatter, 3278
 - ~SimpleFormatter, 3279
 - format, 3279
 - SimpleFormatter, 3279
- decaf::util::logging::SimpleLogger, 3279
 - ~SimpleLogger, 3280
 - debug, 3280

- error, 3280
- fatal, 3280
- info, 3280
- log, 3280
- mark, 3281
- SimpleLogger, 3280
- warn, 3281
- decaf::util::logging::StreamHandler, 3412
 - ~StreamHandler, 3413
 - close, 3414
 - flush, 3414
 - isLoggable, 3414
 - publish, 3414
 - setOutputStream, 3415
 - StreamHandler, 3413
- decaf::util::logging::XMLFormatter, 3793
 - ~XMLFormatter, 3794
 - format, 3794
 - getHead, 3794
 - getTail, 3794
 - XMLFormatter, 3794
- decaf::util::Map, 2305
 - ~Map, 2307
 - clear, 2307
 - containsKey, 2308
 - containsValue, 2308
 - copy, 2309
 - equals, 2310
 - get, 2310, 2311
 - isEmpty, 2312
 - keySet, 2313
 - Map, 2307
 - put, 2313
 - putAll, 2314
 - remove, 2315
 - size, 2316
 - values, 2317
- decaf::util::Map::Entry, 1706
 - ~Entry, 1707
 - Entry, 1707
 - getKey, 1707
 - getValue, 1707
 - setValue, 1707
- decaf::util::PriorityQueue, 2832
 - ~PriorityQueue, 2835
 - add, 2835
 - clear, 2836
 - comparator, 2836
 - iterator, 2836
 - offer, 2836
 - operator=, 2837
 - peek, 2837
 - poll, 2837
 - PriorityQueue, 2834, 2835
 - PriorityQueueIterator, 2839
 - remove, 2838
 - size, 2839
- decaf::util::Properties, 2927
 - ~Properties, 2929
 - clear, 2929
 - clone, 2929
 - copy, 2929
 - defaults, 2936
 - equals, 2929
 - getProperty, 2929, 2930
 - hasProperty, 2930
 - isEmpty, 2930
 - load, 2930, 2931
 - operator=, 2933
 - Properties, 2929
 - propertyNames, 2933
 - remove, 2933
 - setProperty, 2933
 - size, 2934
 - store, 2934
 - toArray, 2935
 - toString, 2935
- decaf::util::Queue, 2948
 - ~Queue, 2949
 - element, 2949
 - offer, 2949
 - peek, 2949
 - poll, 2950
 - remove, 2950
- decaf::util::Random, 2953
 - next, 2954
 - nextBoolean, 2955
 - nextBytes, 2955
 - nextDouble, 2955
 - nextFloat, 2956
 - nextGaussian, 2956
 - nextInt, 2956
 - nextLong, 2957
 - Random, 2954
 - setSeed, 2957
- decaf::util::Set, 3220
 - ~Set, 3220
- decaf::util::StlList, 3357
 - ~StlList, 3362
 - add, 3363
 - addAll, 3364
 - clear, 3364
 - contains, 3365
 - copy, 3365
 - equals, 3365
 - get, 3365
 - indexOf, 3365
 - isEmpty, 3366

- iterator, 3366
- lastIndexOf, 3366
- listIterator, 3367
- remove, 3367, 3368
- set, 3368
- size, 3369
- StlList, 3362
- decaf::util::StlMap, 3369
 - ~StlMap, 3374
 - clear, 3374
 - containsKey, 3374
 - containsValue, 3374
 - copy, 3375
 - equals, 3375
 - get, 3376
 - isEmpty, 3376
 - keySet, 3377
 - lock, 3377
 - notify, 3377
 - notifyAll, 3378
 - put, 3378
 - putAll, 3378
 - remove, 3379
 - size, 3379
 - StlMap, 3373
 - tryLock, 3379
 - unlock, 3380
 - values, 3380
 - wait, 3380, 3381
- decaf::util::StlQueue, 3382
 - ~StlQueue, 3384
 - back, 3384
 - clear, 3385
 - empty, 3385
 - enqueueFront, 3385
 - front, 3385
 - getSafeValue, 3385
 - iterator, 3386
 - lock, 3386
 - notify, 3386
 - notifyAll, 3386
 - pop, 3387
 - push, 3387
 - reverse, 3387
 - size, 3387
 - StlQueue, 3384
 - toArray, 3387
 - tryLock, 3388
 - unlock, 3388
 - wait, 3388, 3389
- decaf::util::StlSet, 3390
 - ~StlSet, 3392
 - add, 3392
 - clear, 3393
 - contains, 3393
 - copy, 3394
 - equals, 3394
 - isEmpty, 3394
 - iterator, 3394
 - remove, 3394
 - size, 3395
 - StlSet, 3392
- decaf::util::StringTokenizer, 3430
 - ~StringTokenizer, 3431
 - countTokens, 3431
 - hasMoreTokens, 3432
 - nextToken, 3432
 - reset, 3432
 - StringTokenizer, 3431
 - toArray, 3433
- decaf::util::Timer, 3543
 - ~Timer, 3545
 - cancel, 3545
 - purge, 3545
 - schedule, 3546–3550
 - scheduleAtFixedRate, 3551–3553
 - Timer, 3545
- decaf::util::TimerTask, 3554
 - ~TimerTask, 3555
 - cancel, 3555
 - decaf::internal::util::TimerTaskHeap, 3556
 - getWhen, 3555
 - isScheduled, 3555
 - scheduledExecutionTime, 3555
 - setScheduledTime, 3556
 - Timer, 3556
 - TimerImpl, 3556
 - TimerTask, 3555
- decaf::util::UUID, 3706
 - ~UUID, 3708
 - clockSequence, 3708
 - compareTo, 3708
 - equals, 3708
 - fromString, 3709
 - getLeastSignificantBits, 3709
 - getMostSignificantBits, 3709
 - nameUUIDFromBytes, 3709
 - node, 3710
 - operator<, 3710
 - operator==, 3710
 - randomUUID, 3711
 - timestamp, 3711
 - toString, 3711
 - UUID, 3708
 - variant, 3711
 - version, 3712
- decaf::util::zip, 140
- decaf::util::zip::Adler32, 663

- ~Adler32, 664
- Adler32, 664
- getValue, 664
- reset, 664
- update, 664, 665
- decaf::util::zip::CheckedInputStream, 1055
 - ~CheckedInputStream, 1056
 - CheckedInputStream, 1056
 - doReadArrayBounded, 1057
 - doReadByte, 1057
 - getChecksum, 1057
 - skip, 1057
- decaf::util::zip::CheckedOutputStream, 1058
 - ~CheckedOutputStream, 1059
 - CheckedOutputStream, 1058
 - doWriteArrayBounded, 1059
 - doWriteByte, 1059
 - getChecksum, 1059
- decaf::util::zip::Checksum, 1059
 - ~Checksum, 1060
 - getValue, 1060
 - reset, 1060
 - update, 1060, 1061
- decaf::util::zip::CRC32, 1420
 - ~CRC32, 1421
 - CRC32, 1421
 - getValue, 1421
 - reset, 1421
 - update, 1421, 1422
- decaf::util::zip::DataFormatException, 1450
 - ~DataFormatException, 1451
 - clone, 1452
 - DataFormatException, 1450, 1451
- decaf::util::zip::Deflater, 1595
 - ~Deflater, 1597
 - BEST_COMPRESSION, 1603
 - BEST_SPEED, 1603
 - DEFAULT_COMPRESSION, 1603
 - DEFAULT_STRATEGY, 1604
 - deflate, 1598
 - DEFLATED, 1604
 - Deflater, 1597
 - end, 1599
 - FILTERED, 1604
 - finish, 1599
 - finished, 1599
 - getAdler, 1599
 - getBytesRead, 1599
 - getBytesWritten, 1600
 - HUFFMAN_ONLY, 1604
 - needsInput, 1600
 - NO_COMPRESSION, 1604
 - reset, 1600
 - setDictionary, 1600, 1601
 - setInput, 1601, 1602
 - setLevel, 1603
 - setStrategy, 1603
- decaf::util::zip::DeflaterOutputStream, 1604
 - ~DeflaterOutputStream, 1607
 - buf, 1608
 - close, 1607
 - DEFAULT_BUFFER_SIZE, 1608
 - deflate, 1607
 - deflater, 1608
 - DeflaterOutputStream, 1606
 - doWriteArrayBounded, 1607
 - doWriteByte, 1607
 - finish, 1607
 - isDone, 1608
 - ownDeflater, 1608
- decaf::util::zip::Inflater, 1894
 - ~Inflater, 1896
 - end, 1896
 - finish, 1896
 - finished, 1896
 - getAdler, 1896
 - getBytesRead, 1896
 - getBytesWritten, 1897
 - getRemaining, 1897
 - inflate, 1897, 1898
 - Inflater, 1896
 - needsDictionary, 1898
 - needsInput, 1899
 - reset, 1899
 - setDictionary, 1899, 1900
 - setInput, 1900, 1901
- decaf::util::zip::InflaterInputStream, 1902
 - ~InflaterInputStream, 1905
 - atEOF, 1909
 - available, 1906
 - buff, 1909
 - close, 1906
 - DEFAULT_BUFFER_SIZE, 1909
 - doReadArrayBounded, 1906
 - doReadByte, 1906
 - fill, 1906
 - inflator, 1909
 - InflaterInputStream, 1904, 1905
 - length, 1909
 - mark, 1907
 - markSupported, 1907
 - ownInflater, 1909
 - reset, 1907
 - skip, 1908
- decaf::util::zip::ZipException, 3796
 - ~ZipException, 3797
 - clone, 3798
 - ZipException, 3796, 3797

- DECAF_API
 - decaf/util/Config.h, 3890
- DECAF_CATCH_EXCEPTION_CONVERT
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3857
- DECAF_CATCH_NOTHROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3857
- DECAF_CATCH_RETHROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3857
- DECAF_CATCHALL_NOTHROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3858
- DECAF_CATCHALL_THROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3858
- DECAF_UNUSED
 - decaf/util/Config.h, 3890
- DecafRuntime
 - decaf::internal::DecafRuntime, 1563
- decode
 - decaf::internal::net::URLEncoderDecoder,
3673
 - decaf::lang::Byte, 887
 - decaf::lang::Integer, 1946
 - decaf::lang::Long, 2272
 - decaf::lang::Short, 3223
 - decaf::net::URLDecoder, 3700
- decreaseUsage
 - activemq::util::MemoryUsage, 2356
 - activemq::util::Usage, 3702
- decrementAndGet
 - decaf::util::concurrent::atomic::AtomicInteger,
682
- DedicatedTaskRunner
 - activemq::threads::DedicatedTaskRunner,
1565
- DEF_MEM_LEVEL
 - zutil.h, 4217
- DEF_WBITS
 - zutil.h, 4217
- DEFAULT_BUFFER_SIZE
 - decaf::util::zip::DeflaterOutputStream,
1608
 - decaf::util::zip::InflaterInputStream, 1909
- DEFAULT_COMPRESSION
 - decaf::util::zip::Deflater, 1603
- DEFAULT_DURABLE_TOPIC_-
PREFETCH
 - activemq::core::policies::DefaultPrefetchPolicy,
1569
- DEFAULT_MAX_BLOCK_SIZE
 - decaf::util::concurrent::ThreadPool, 3536
- DEFAULT_MAX_POOL_SIZE
 - decaf::util::concurrent::ThreadPool, 3536
- DEFAULT_MESSAGE_SIZE
 - activemq::commands::Message, 2374
- DEFAULT_ORDERED_TARGET
 - activemq::commands::ActiveMQDestination,
288
- DEFAULT_PRIORITY
 - activemq::cmsutil::CmsTemplate, 1096
- DEFAULT_QUEUE_BROWSER_-
PREFETCH
 - activemq::core::policies::DefaultPrefetchPolicy,
1569
- DEFAULT_QUEUE_PREFETCH
 - activemq::core::policies::DefaultPrefetchPolicy,
1569
- DEFAULT_STRATEGY
 - decaf::util::zip::Deflater, 1604
- DEFAULT_TIME_TO_LIVE
 - activemq::cmsutil::CmsTemplate, 1096
- DEFAULT_TOPIC_PREFETCH
 - activemq::core::policies::DefaultPrefetchPolicy,
1569
- DEFAULT_URI
 - activemq::core::ActiveMQConnectionFactory,
261
- DEFAULT_VERSION
 - activemq::wireformat::openwire::OpenWireFormat,
2712
- DefaultPrefetchPolicy
 - activemq::core::policies::DefaultPrefetchPolicy,
1567
- DefaultRedeliveryPolicy
 - activemq::core::policies::DefaultRedeliveryPolicy,
1570
- defaults
 - decaf::util::Properties, 2936
- DefaultServerSocketFactory
 - decaf::internal::net::DefaultServerSocketFactory,
1575
- DefaultSocketFactory
 - decaf::internal::net::DefaultSocketFactory,
1579
- DefaultSSLContext
 - decaf::internal::net::ssl::DefaultSSLContext,
1582
- DefaultSSLServerSocketFactory
 - decaf::internal::net::ssl::DefaultSSLServerSocketFactory,
1584
- DefaultSSLSocketFactory
 - decaf::internal::net::ssl::DefaultSSLSocketFactory,
1589
- deflate
 - decaf::util::zip::Deflater, 1598

- decaf::util::zip::DeflaterOutputStream, 1607
- deflate.h
 - _dist_code, 4202
 - _length_code, 4202
 - _tr_tally_dist, 4201
 - _tr_tally_lit, 4201
 - BL_CODES, 4201
 - BUSY_STATE, 4202
 - Code, 4202
 - COMMENT_STATE, 4202
 - ct_data, 4202
 - d_code, 4202
 - D_CODES, 4202
 - Dad, 4202
 - deflate_state, 4202
 - EXTRA_STATE, 4202
 - FINISH_STATE, 4202
 - Freq, 4202
 - GZIP, 4202
 - HCRC_STATE, 4202
 - HEAP_SIZE, 4202
 - INIT_STATE, 4202
 - IPos, 4202
 - L_CODES, 4202
 - Len, 4202
 - LENGTH_CODES, 4202
 - LITERALS, 4202
 - MAX_BITS, 4202
 - MAX_DIST, 4202
 - max_insert_length, 4202
 - MIN_LOOKAHEAD, 4202
 - NAME_STATE, 4202
 - OF, 4202
 - Pos, 4202
 - Posf, 4202
 - put_byte, 4202
 - static_tree_desc, 4202
 - tree_desc, 4202
 - WIN_INIT, 4202
- deflate_state
 - deflate.h, 4202
- DEFLATED
 - decaf::util::zip::Deflater, 1604
- deflateInit
 - zlib.h, 4214
- deflateInit2
 - zlib.h, 4214
- Deflater
 - decaf::util::zip::Deflater, 1597
- deflater
 - decaf::util::zip::DeflaterOutputStream, 1608
- DeflaterOutputStream
 - decaf::util::zip::DeflaterOutputStream, 1606
- deleteIfCancelled
 - decaf::internal::util::TimerTaskHeap, 3557
- deliverAcks
 - activemq::core::ActiveMQConsumer, 272
 - activemq::core::ActiveMQSession, 475
- DELIVERY_MODE
 - cms::DeliveryMode, 1610
- deliverySequenceId
 - activemq::commands::MessageDispatchNotification, 2466
- depth
 - internal_state, 1985
- dequeue
 - activemq::core::ActiveMQConsumer, 273
 - activemq::core::MessageDispatchChannel, 2433
- dequeueNoWait
 - activemq::core::MessageDispatchChannel, 2433
- deQueueTask
 - decaf::util::concurrent::ThreadPool, 3533
- descriptor
 - decaf::io::FileDescriptor, 1769
- destination
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2869
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2972
 - activemq::commands::ConsumerControl, 1306
 - activemq::commands::ConsumerInfo, 1365
 - activemq::commands::DestinationInfo, 1618
 - activemq::commands::JournalQueueAck, 2017
 - activemq::commands::JournalTopicAck, 2045
 - activemq::commands::Message, 2374
 - activemq::commands::MessageAck, 2399
 - activemq::commands::MessageDispatch, 2431
 - activemq::commands::MessageDispatchNotification, 2466
 - activemq::commands::MessagePull, 2568
 - activemq::commands::ProducerInfo, 2902
 - activemq::commands::SubscriptionInfo, 3438
- DESTINATION_ADD_OPERATION
 - activemq::core::ActiveMQConstants, 267
- DESTINATION_REMOVE_OPERATION
 - activemq::core::ActiveMQConstants, 267
- DestinationActions

- activemq::core::ActiveMQConstants, 267
- DestinationInfo
 - activemq::commands::DestinationInfo, 1615
- DestinationInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1631
 - activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1619
 - activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1623
 - activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1627
 - activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1639
 - activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1635
- DestinationOption
 - activemq::core::ActiveMQConstants, 267
- DestinationType
 - cms::Destination, 1611
- destOptionMap
 - activemq::core::ActiveMQConstants::StaticInitializer, 3357
- destOptions
 - activemq::core::ActiveMQConstants::StaticInitializer, 3357
- destroy
 - activemq::cmsutil::CmsAccessor, 1070
 - activemq::cmsutil::CmsDestinationAccessor, 1073
 - activemq::cmsutil::CmsTemplate, 1087
 - activemq::cmsutil::DestinationResolver, 1642
 - activemq::cmsutil::DynamicDestinationResolver, 1705
 - activemq::cmsutil::ResourceLifecycleManager, 3076
 - activemq::commands::ActiveMQTempQueue, 551
 - activemq::commands::ActiveMQTempTopic, 579
 - cms::TemporaryQueue, 3517
 - cms::TemporaryTopic, 3518
 - decaf::internal::util::concurrent::ConditionImpl, 1164
 - decaf::internal::util::concurrent::MutexImpl, 2610
 - decaf::util::logging::LogWriter, 2266
- destroyDestination
 - activemq::core::ActiveMQConnection, 240
- destroyMarshalers
 - activemq::wireformat::openwire::OpenWireFormat, 2704
- destroyResources
 - decaf::internal::util::ResourceLifecycleManager, 3073
- DICT
 - inflate.h, 4206
- DISC
 - inflate.h, 4206
- DISCINFO
 - inflate.h, 4206
- DISCONNECT
 - decaf::lang::Character, 1022
- DISCONNECTED
 - gz_state, 1851
- DISCONNECTED
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- DiscoveryEvent
 - activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1645
- DiscoveryEventMarshaller
 - activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1664
 - activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1652
 - activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1656
 - activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1660
 - activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1668
 - activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1648
- dispatch
 - activemq::core::ActiveMQConsumer, 273
 - activemq::core::ActiveMQSession, 476
 - activemq::core::Dispatcher, 1672
- DispatchAsync
 - activemq::commands::ConsumerInfo, 1365
 - activemq::commands::ProducerInfo, 2902
- DispatchData
 - activemq::core::DispatchData, 1671
- DIST
 - inflate.h, 4206
- distbits
 - inflate_state, 1893
- distcode
 - inflate_state, 1893
- DISTEXT
 - inflate.h, 4206
- DISTS
 - inftrees.h, 4207
- dl
 - ct_data_s, 1423
- dmax
 - inflate_state, 1893
- doAppendChar

- decaf::io::Writer, 3758
- doAppendCharSequence
 - decaf::io::Writer, 3759
- doAppendCharSequenceStartEnd
 - decaf::io::Writer, 3759
- doClose
 - activemq::core::ActiveMQConsumer, 273
- doCreateComposite
 - activemq::transport::failover::FailoverTransportFactory, 1765
 - activemq::transport::mock::MockTransportFactory, 2603
 - activemq::transport::tcp::SslTransportFactory, 3350
 - activemq::transport::tcp::TcpTransportFactory, 3515
- doInCms
 - activemq::cmsutil::CmsTemplate::ProducerExtension, 2868
 - activemq::cmsutil::CmsTemplate::ReceiveExtension, 2971
 - activemq::cmsutil::CmsTemplate::SendExecution, 3136
 - activemq::cmsutil::ProducerCallback, 2867
 - activemq::cmsutil::SessionCallback, 3161
- DONE
 - inflate.h, 4206
- done
 - gz_header_s, 1849
- doReadArray
 - decaf::io::FilterInputStream, 1774
 - decaf::io::InputStream, 1912
 - decaf::io::Reader, 2962
- doReadArrayBounded
 - activemq::io::LoggingInputStream, 2250
 - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2697
 - decaf::internal::net::tcp::TcpSocketInputStream, 3508
 - decaf::io::BlockingByteArrayInputStream, 773
 - decaf::io::BufferedInputStream, 864
 - decaf::io::ByteArrayInputStream, 948
 - decaf::io::FilterInputStream, 1774
 - decaf::io::InputStream, 1912
 - decaf::io::InputStreamReader, 1921
 - decaf::io::PushbackInputStream, 2943
 - decaf::io::Reader, 2962
 - decaf::util::zip::CheckedInputStream, 1057
 - decaf::util::zip::InflaterInputStream, 1906
- doReadByte
 - activemq::io::LoggingInputStream, 2250
 - decaf::internal::io::StandardInputStream, 3353
 - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2697
 - decaf::internal::net::tcp::TcpSocketInputStream, 3508
 - decaf::io::BlockingByteArrayInputStream, 773
 - decaf::io::BufferedInputStream, 865
 - decaf::io::ByteArrayInputStream, 948
 - decaf::io::FilterInputStream, 1774
 - decaf::io::InputStream, 1912
 - decaf::io::PushbackInputStream, 2943
 - decaf::util::zip::CheckedInputStream, 1057
 - decaf::util::zip::InflaterInputStream, 1906
 - doReadChar
 - decaf::io::Reader, 2962
 - doReadCharBuffer
 - decaf::io::Reader, 2962
 - doReadDouble
 - decaf::lang::Double, 1674
 - DOUBLE_TYPE
 - activemq::util::PrimitiveValueNode, 2821
 - DoubleArrayBuffer
 - decaf::internal::nio::DoubleArrayBuffer, 1686, 1687
 - DoubleBuffer
 - decaf::nio::DoubleBuffer, 1695
 - doubleToLongBits
 - decaf::lang::Double, 1676
 - doubleToRawLongBits
 - decaf::lang::Double, 1676
 - doubleValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2815
 - decaf::lang::Byte, 888
 - decaf::lang::Character, 1023
 - decaf::lang::Double, 1677
 - decaf::lang::Float, 1784
 - decaf::lang::Integer, 1946
 - decaf::lang::Long, 2272
 - decaf::lang::Number, 2654
 - decaf::lang::Short, 3224
 - decaf::util::concurrent::atomic::AtomicInteger, 682
- doUnmarshal
 - activemq::wireformat::openwire::OpenWireFormat, 2704
- doWriteArray
 - decaf::io::BufferedOutputStream, 868
 - decaf::io::FilterOutputStream, 1778
 - decaf::io::OutputStream, 2720

- decaf::io::Writer, 3759
- doWriteArrayBounded
 - activemq::io::LoggingOutputStream, 2252
 - decaf::internal::io::StandardErrorOutputStream, 3352
 - decaf::internal::io::StandardOutputStream, 3354
 - decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2699
 - decaf::internal::net::tcp::TcpSocketOutputStream, 3510
 - decaf::io::BufferedOutputStream, 868
 - decaf::io::ByteArrayOutputStream, 952
 - decaf::io::DataOutputStream, 1475
 - decaf::io::FilterOutputStream, 1779
 - decaf::io::OutputStream, 2721
 - decaf::io::OutputStreamWriter, 2727
 - decaf::io::Writer, 3759
 - decaf::util::zip::CheckedOutputStream, 1059
 - decaf::util::zip::DeflaterOutputStream, 1607
- doWriteByte
 - activemq::io::LoggingOutputStream, 2252
 - decaf::internal::io::StandardErrorOutputStream, 3352
 - decaf::internal::io::StandardOutputStream, 3355
 - decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2699
 - decaf::internal::net::tcp::TcpSocketOutputStream, 3510
 - decaf::io::BufferedOutputStream, 868
 - decaf::io::ByteArrayOutputStream, 952
 - decaf::io::DataOutputStream, 1475
 - decaf::io::FilterOutputStream, 1779
 - decaf::io::OutputStream, 2721
 - decaf::util::zip::CheckedOutputStream, 1059
 - decaf::util::zip::DeflaterOutputStream, 1607
- doWriteChar
 - decaf::io::Writer, 3759
- doWriteString
 - decaf::io::Writer, 3759
- doWriteStringBounded
 - decaf::io::Writer, 3759
- doWriteVector
 - decaf::io::Writer, 3759
- drainPermits
 - decaf::util::concurrent::Semaphore, 3131
- drainTo
 - decaf::util::concurrent::BlockingQueue, 777, 778
- decaf::util::concurrent::SynchronousQueue, 3481
- droppable
 - activemq::commands::Message, 2374
- dummy
 - internal_state, 1985
- duplexConnection
 - activemq::commands::BrokerInfo, 831
- duplicate
 - decaf::internal::nio::ByteBuffer, 931
 - decaf::internal::nio::CharArrayBuffer, 1034
 - decaf::internal::nio::DoubleArrayBuffer, 1689
 - decaf::internal::nio::FloatArrayBuffer, 1797
 - decaf::internal::nio::IntArrayBuffer, 1928
 - decaf::internal::nio::LongArrayBuffer, 2288
 - decaf::internal::nio::ShortArrayBuffer, 3235
 - decaf::nio::ByteBuffer, 963
 - decaf::nio::CharBuffer, 1044
 - decaf::nio::DoubleBuffer, 1697
 - decaf::nio::FloatBuffer, 1805
 - decaf::nio::IntBuffer, 1935
 - decaf::nio::LongBuffer, 2296
 - decaf::nio::ShortBuffer, 3243
- DUPS_OK_ACKNOWLEDGE
 - cms::Session, 3152
- dyn_dtree
 - internal_state, 1985
- dyn_ltree
 - internal_state, 1985
- dyn_tree
 - tree_desc_s, 3648
- DYN_TREES
 - zutil.h, 4217
- dynamicCast
 - decaf::lang::Pointer, 2760
- DynamicDestinationResolver
 - activemq::cmsutil::DynamicDestinationResolver, 1705
- E
 - decaf::lang::Math, 2354
- element
 - decaf::util::AbstractQueue, 160
 - decaf::util::Queue, 2949
- empty
 - decaf::util::StlQueue, 3385
- encode
 - decaf::net::URLEncoder, 3701
- encodeOthers
 - decaf::internal::net::URLEncoderDecoder, 3673
- end

- decaf::util::zip::Deflater, 1599
- decaf::util::zip::Inflater, 1896
- ENOUGH
 - inftrees.h, 4207
- ENOUGH_DISTS
 - inftrees.h, 4207
- ENOUGH_LENS
 - inftrees.h, 4207
- enqueue
 - activemq::core::MessageDispatchChannel, 2434
- enqueueFirst
 - activemq::core::MessageDispatchChannel, 2434
- enqueueFront
 - decaf::util::StlQueue, 3385
- enqueueUsage
 - activemq::util::MemoryUsage, 2356
 - activemq::util::Usage, 3702
- ensureCreated
 - decaf::net::ServerSocket, 3142
 - decaf::net::Socket, 3288
- entering
 - decaf::util::logging::Logger, 2242
- Entry
 - decaf::util::Map::Entry, 1707
- eof
 - gz_state, 1851
- EOFException
 - decaf::io::EOFException, 1708, 1709
- equals
 - activemq::commands::ActiveMQBlobMessage, 168
 - activemq::commands::ActiveMQBytesMessage, 199
 - activemq::commands::ActiveMQDestination, 283
 - activemq::commands::ActiveMQMapMessage, 320
 - activemq::commands::ActiveMQMessage, 354
 - activemq::commands::ActiveMQMessageTemplate, 383
 - activemq::commands::ActiveMQObjectMessage, 398
 - activemq::commands::ActiveMQQueue, 437
 - activemq::commands::ActiveMQStreamMessage, 489
 - activemq::commands::ActiveMQTempDestination, 525
 - activemq::commands::ActiveMQTempQueue, 551
 - activemq::commands::ActiveMQTempTopic, 579
 - activemq::commands::ActiveMQTextMessage, 607
 - activemq::commands::ActiveMQTopic, 635
 - activemq::commands::BaseCommand, 696
 - activemq::commands::BaseDataStructure, 766
 - activemq::commands::BooleanExpression, 787
 - activemq::commands::BrokerId, 800
 - activemq::commands::BrokerInfo, 826
 - activemq::commands::ConnectionControl, 1173
 - activemq::commands::ConnectionError, 1202
 - activemq::commands::ConnectionId, 1232, 1233
 - activemq::commands::ConnectionInfo, 1260
 - activemq::commands::ConsumerControl, 1304
 - activemq::commands::ConsumerId, 1332
 - activemq::commands::ConsumerInfo, 1360
 - activemq::commands::ControlCommand, 1391
 - activemq::commands::DataArrayResponse, 1424
 - activemq::commands::DataResponse, 1478
 - activemq::commands::DataStructure, 1556
 - activemq::commands::DestinationInfo, 1615
 - activemq::commands::DiscoveryEvent, 1645
 - activemq::commands::ExceptionResponse, 1721
 - activemq::commands::FlushCommand, 1813
 - activemq::commands::IntegerResponse, 1957
 - activemq::commands::JournalQueueAck, 2015
 - activemq::commands::JournalTopicAck, 2042
 - activemq::commands::JournalTrace, 2071
 - activemq::commands::JournalTransaction, 2097
 - activemq::commands::KeepAliveInfo, 2124
 - activemq::commands::LastPartialCommand, 2157
 - activemq::commands::LocalTransactionId, 2202
 - activemq::commands::Message, 2363
 - activemq::commands::MessageAck, 2395

- activemq::commands::MessageDispatch, 2428
- activemq::commands::MessageDispatchNotification, 2464
- activemq::commands::MessageId, 2496, 2497
- activemq::commands::MessagePull, 2565
- activemq::commands::NetworkBridgeFilter, 2615
- activemq::commands::PartialCommand, 2729
- activemq::commands::ProducerAck, 2841
- activemq::commands::ProducerId, 2871, 2872
- activemq::commands::ProducerInfo, 2899
- activemq::commands::RemoveInfo, 2989
- activemq::commands::RemoveSubscriptionInfo, 3016
- activemq::commands::ReplayCommand, 3044
- activemq::commands::Response, 3078
- activemq::commands::SessionId, 3163, 3164
- activemq::commands::SessionInfo, 3190
- activemq::commands::ShutdownInfo, 3251
- activemq::commands::SubscriptionInfo, 3435
- activemq::commands::TransactionId, 3571
- activemq::commands::TransactionInfo, 3596
- activemq::commands::WireFormatInfo, 3721
- activemq::commands::XATransactionId, 3766, 3767
- decaf::lang::Boolean, 783
- decaf::lang::Byte, 888
- decaf::lang::Character, 1023
- decaf::lang::Comparable, 1126
- decaf::lang::Double, 1677
- decaf::lang::Float, 1784
- decaf::lang::Integer, 1946, 1947
- decaf::lang::Long, 2272
- decaf::lang::Short, 3224
- decaf::net::URI, 3665
- decaf::nio::ByteBuffer, 964
- decaf::nio::CharBuffer, 1045
- decaf::nio::DoubleBuffer, 1697
- decaf::nio::FloatBuffer, 1805
- decaf::nio::IntBuffer, 1935
- decaf::nio::LongBuffer, 2296
- decaf::nio::ShortBuffer, 3243
- decaf::security::cert::Certificate, 1008
- decaf::security::Principal, 2831
- decaf::util::AbstractCollection, 149
- decaf::util::Collection, 1102
- decaf::util::concurrent::ConcurrentStlMap, 1147
- decaf::util::concurrent::SynchronousQueue, 3482
- decaf::util::concurrent::TimeUnit, 3562
- decaf::util::Date, 1561
- decaf::util::logging::Level, 2188
- decaf::util::Map, 2310
- decaf::util::Properties, 2929
- decaf::util::StlList, 3365
- decaf::util::StlMap, 3375
- decaf::util::StlSet, 3394
- decaf::util::UUID, 3708
- err
 - decaf::io::FileDescriptor, 1769
 - gz_state, 1851
- ERR_MSG
 - zutil.h, 4217
- ERR_RETURN
 - zutil.h, 4217
- Error
 - decaf::util::logging, 140
- error
 - decaf::util::logging::ErrorManager, 1711
 - decaf::util::logging::SimpleLogger, 3280
- ERROR_CMD
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- ErrorManager
 - decaf::util::logging::ErrorManager, 1711
- Exception
 - decaf::lang::Exception, 1713, 1714
- exception
 - activemq::commands::ConnectionError, 1204
 - activemq::commands::ExceptionResponse, 1722
- ExceptionResponse
 - activemq::commands::ExceptionResponse, 1721
- ExceptionResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::ExceptionResponse, 1744
 - activemq::wireformat::openwire::marshal::v2::ExceptionResponse, 1728
 - activemq::wireformat::openwire::marshal::v3::ExceptionResponse, 1732
 - activemq::wireformat::openwire::marshal::v4::ExceptionResponse, 1740
 - activemq::wireformat::openwire::marshal::v5::ExceptionResponse, 1736
 - activemq::wireformat::openwire::marshal::v6::ExceptionResponse, 1724
- exclusive

- activemq::commands::ActiveMQDestination, 288
- activemq::commands::ConsumerInfo, 1365
- execute
 - activemq::cmsutil::CmsTemplate, 1087, 1088
 - activemq::core::ActiveMQSessionExecutor, 484
 - decaf::util::concurrent::Executor, 1750
- executeFirst
 - activemq::core::ActiveMQSessionExecutor, 484
- ExecutionException
 - decaf::util::concurrent::ExecutionException, 1747, 1748
- exit
 - activemq::commands::ConnectionControl, 1176
- exiting
 - decaf::util::logging::Logger, 2242
- EXLEN
 - inflate.h, 4205
- expiration
 - activemq::commands::Message, 2374
- EXTRA
 - inflate.h, 4206
- extra
 - gz_header_s, 1849
 - inflate_state, 1893
- extra_len
 - gz_header_s, 1849
- extra_max
 - gz_header_s, 1849
- EXTRA_STATE
 - deflate.h, 4202
- F_OPEN
 - zutil.h, 4217
- failIfReadOnlyBody
 - activemq::commands::ActiveMQMessageTemplate, 384
- failIfReadOnlyProperties
 - activemq::commands::ActiveMQMessageTemplate, 384
- failIfWriteOnlyBody
 - activemq::commands::ActiveMQMessageTemplate, 384
- FailoverTransport
 - activemq::transport::failover::FailoverTransport, 1755
- FailoverTransportListener
 - activemq::transport::failover::FailoverTransportListener, 1767
- activemq::transport::failover::FailoverTransportListener, 1767
- FAR
 - zconf.h, 4211
- Fatal
 - decaf::util::logging, 140
- fatal
 - decaf::util::logging::SimpleLogger, 3280
- faultTolerant
 - activemq::commands::ConnectionControl, 1176
 - activemq::commands::ConnectionInfo, 1263
- faultTolerantConfiguration
 - activemq::commands::BrokerInfo, 831
- fc
 - ct_data_s, 1423
- fd
 - decaf::net::SocketImpl, 3313
 - gz_state, 1851
- FileDescriptor
 - decaf::io::FileDescriptor, 1769
- FileName
 - activemq::commands::BrokerError::StackTraceElement, 3351
- fill
 - decaf::util::zip::InflaterInputStream, 1906
- FILTERED
 - decaf::util::zip::Deflater, 1604
- FilterInputStream
 - decaf::io::FilterInputStream, 1773
- FilterOutputStream
 - decaf::io::FilterOutputStream, 1778
- find
 - decaf::internal::util::TimerTaskHeap, 3557
- findFactory
 - activemq::transport::TransportRegistry, 3646
 - activemq::wireformat::WireFormatRegistry, 3753
- fine
 - decaf::util::logging::Level, 2189
- decaf::util::logging::Logger, 2242
- FINER
 - decaf::util::logging::Level, 2189
- decaf::util::logging::Logger, 2243
- FINEST
 - decaf::util::logging::Level, 2189
- finest
 - decaf::util::logging::Logger, 2243
- finish
 - decaf::util::zip::Deflater, 1599

- decaf::util::zip::DeflaterOutputStream, 1607
- decaf::util::zip::Inflater, 1896
- FINISH_STATE
 - deflate.h, 4202
- finished
 - decaf::util::zip::Deflater, 1599
 - decaf::util::zip::Inflater, 1896
- fire
 - activemq::core::ActiveMQConnection, 241
 - activemq::core::ActiveMQSession, 476
 - activemq::transport::TransportFilter, 3638
- fireCommand
 - activemq::transport::mock::MockTransport, 2595
- fireException
 - activemq::transport::mock::MockTransport, 2595
- firstMessageId
 - activemq::commands::MessageAck, 2399
- firstNakNumber
 - activemq::commands::ReplayCommand, 3046
- FLAGS
 - inflate.h, 4205
- flags
 - inflate_state, 1893
- flip
 - decaf::nio::Buffer, 858
- Float
 - decaf::lang::Float, 1782, 1783
- FLOAT_TYPE
 - activemq::util::PrimitiveValueNode, 2821
- FloatArrayBuffer
 - decaf::internal::nio::FloatArrayBuffer, 1794, 1795
- FloatBuffer
 - decaf::nio::FloatBuffer, 1802
- floatToIntBits
 - decaf::lang::Float, 1784
- floatToRawIntBits
 - decaf::lang::Float, 1785
- floatValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2815
 - decaf::lang::Byte, 888
 - decaf::lang::Character, 1023
 - decaf::lang::Double, 1677
 - decaf::lang::Float, 1785
 - decaf::lang::Integer, 1947
 - decaf::lang::Long, 2273
 - decaf::lang::Number, 2654
 - decaf::lang::Short, 3224
 - decaf::util::concurrent::atomic::AtomicInteger, 682
- floor
- decaf::lang::Math, 2344
- flush
 - activemq::commands::ConsumerControl, 1306
 - decaf::internal::io::StandardErrorOutputStream, 3352
 - decaf::internal::io::StandardOutputStream, 3355
 - decaf::io::BufferedOutputStream, 868
 - decaf::io::FilterOutputStream, 1779
 - decaf::io::Flushable, 1811
 - decaf::io::OutputStream, 2721
 - decaf::io::OutputStreamWriter, 2727
 - decaf::util::logging::Handler, 1853
 - decaf::util::logging::StreamHandler, 3414
- FLUSH_FAILURE
 - decaf::util::logging::ErrorManager, 1711
- FlushCommand
 - activemq::commands::FlushCommand, 1813
- FlushCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::FlushCommand, 1832
 - activemq::wireformat::openwire::marshal::v2::FlushCommand, 1820
 - activemq::wireformat::openwire::marshal::v3::FlushCommand, 1824
 - activemq::wireformat::openwire::marshal::v4::FlushCommand, 1828
 - activemq::wireformat::openwire::marshal::v5::FlushCommand, 1836
 - activemq::wireformat::openwire::marshal::v6::FlushCommand, 1816
- format
 - decaf::util::logging::Formatter, 1839
 - decaf::util::logging::SimpleFormatter, 3279
 - decaf::util::logging::XMLFormatter, 3794
- FORMAT_FAILURE
 - decaf::util::logging::ErrorManager, 1711
- formatId
 - activemq::commands::XATransactionId, 3769
- formatMessage
 - decaf::util::logging::Formatter, 1839
- Freq
 - deflate.h, 4202
- freq
 - ct_data_s, 1423
- fromStream
 - activemq::wireformat::stomp::StompFrame, 3400

- fromString
 - decaf::util::UUID, 3709
- front
 - decaf::util::StlQueue, 3385
- FutureResponse
 - activemq::transport::correlator::FutureResponse, 1844
- GeneralSecurityException
 - decaf::security::GeneralSecurityException, 1845, 1846
- generateId
 - activemq::util::IdGenerator, 1862
- generation
 - decaf::util::concurrent::ConditionHandle, 1163
- GENERIC_FAILURE
 - decaf::util::logging::ErrorManager, 1711
- GenericResource
 - decaf::internal::util::GenericResource, 1848
- get
 - decaf::internal::nio::ByteBuffer, 932
 - decaf::internal::nio::CharArrayBuffer, 1034
 - decaf::internal::nio::DoubleArrayBuffer, 1690
 - decaf::internal::nio::FloatArrayBuffer, 1797, 1798
 - decaf::internal::nio::IntArrayBuffer, 1928
 - decaf::internal::nio::LongArrayBuffer, 2288
 - decaf::internal::nio::ShortArrayBuffer, 3236
 - decaf::internal::util::ByteArrayAdapter, 901
 - decaf::lang::ArrayPointer, 673
 - decaf::lang::Pointer, 2760
 - decaf::nio::ByteBuffer, 964, 965
 - decaf::nio::CharBuffer, 1045, 1046
 - decaf::nio::DoubleBuffer, 1698, 1699
 - decaf::nio::FloatBuffer, 1805, 1806
 - decaf::nio::IntBuffer, 1936, 1937
 - decaf::nio::LongBuffer, 2296, 2297
 - decaf::nio::ShortBuffer, 3244, 3245
 - decaf::util::concurrent::atomic::AtomicBoolean, 679
 - decaf::util::concurrent::atomic::AtomicInteger, 683
 - decaf::util::concurrent::atomic::AtomicReference, 689
 - decaf::util::concurrent::ConcurrentStlMap, 1147, 1148
 - decaf::util::concurrent::Future, 1842
 - decaf::util::List, 2193
 - decaf::util::Map, 2310, 2311
 - decaf::util::StlList, 3365
 - decaf::util::StlMap, 3376
- getAckHandler
 - activemq::commands::Message, 2364
- getAckMode
 - activemq::commands::SessionInfo, 3190
- getAcknowledgeMode
 - activemq::cmsutil::PooledSession, 2775
 - activemq::core::ActiveMQSession, 476
 - cms::Session, 3158
- getAckType
 - activemq::commands::MessageAck, 2396
- getAdditionalPredicate
 - activemq::commands::ConsumerInfo, 1360, 1361
- getAddress
 - decaf::net::InetAddress, 1887
- getAdler
 - decaf::util::zip::Deflater, 1599
 - decaf::util::zip::Inflater, 1896
- getAlgorithm
 - decaf::security::Key, 2150
- getAndAdd
 - decaf::util::concurrent::atomic::AtomicInteger, 683
- getAndDecrement
 - decaf::util::concurrent::atomic::AtomicInteger, 683
- getAndIncrement
 - decaf::util::concurrent::atomic::AtomicInteger, 683
- getAndSet
 - decaf::util::concurrent::atomic::AtomicBoolean, 679
 - decaf::util::concurrent::atomic::AtomicInteger, 683
 - decaf::util::concurrent::atomic::AtomicReference, 689
- getAnonymousLogger
 - decaf::util::logging::Logger, 2243
- getAprPool
 - decaf::internal::AprPool, 669
- getArrival
 - activemq::commands::Message, 2364
- getAuthority
 - decaf::internal::net::URIType, 3692
 - decaf::net::URI, 3665
- getBacklog
 - decaf::util::concurrent::ThreadPool, 3533
- getBackOffMultiplier
 - activemq::core::policies::DefaultRedeliveryPolicy, 1571
 - activemq::core::RedeliveryPolicy, 2974
 - activemq::transport::failover::FailoverTransport, 1756

- getBackup
 - activemq::transport::failover::BackupTransportPool, 1260
 - 693
- getBackupPoolSize
 - activemq::transport::failover::BackupTransportPool, 1615, 1616
 - 693
 - activemq::transport::failover::FailoverTransport, 2899, 2900
- getBasicConstraints
 - decaf::security::cert::X509Certificate, 3764
- getBlockSize
 - decaf::util::concurrent::ThreadPool, 3533
- getBody
 - activemq::wireformat::stomp::StompFrame, 3400
- getBodyBytes
 - activemq::commands::ActiveMQBytesMessage, 199
 - cms::BytesMessage, 982
- getBodyLength
 - activemq::commands::ActiveMQBytesMessage, 199
 - activemq::wireformat::stomp::StompFrame, 3400
 - cms::BytesMessage, 983
- getBool
 - activemq::util::PrimitiveList, 2791
 - activemq::util::PrimitiveMap, 2801
 - activemq::util::PrimitiveValueNode, 2824
- getBoolean
 - activemq::commands::ActiveMQMapMessage, 320
 - cms::MapMessage, 2320
- getBooleanProperty
 - activemq::commands::ActiveMQMessageTemplate, 384
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2559
 - cms::Message, 2380
- getBranchQualifier
 - activemq::commands::XATransactionId, 3767
- getBrokerId
 - activemq::commands::BrokerInfo, 827
- getBrokerInTime
 - activemq::commands::Message, 2364
- getBrokerName
 - activemq::commands::BrokerInfo, 827
 - activemq::commands::DiscoveryEvent, 1645, 1646
- getBrokerOutTime
 - activemq::commands::Message, 2364
- getBrokerPath
 - activemq::commands::ConnectionInfo, 1260
 - activemq::commands::ConsumerInfo, 1361
 - activemq::commands::DestinationInfo, 1615, 1616
 - activemq::commands::Message, 2364
 - activemq::commands::ProducerInfo, 2899, 2900
- getBrokerSequenceId
 - activemq::commands::MessageId, 2497
- getBrokerUploadUrl
 - activemq::commands::BrokerInfo, 827
- getBrokerURL
 - activemq::commands::BrokerInfo, 827
 - activemq::core::ActiveMQConnection, 241
 - activemq::core::ActiveMQConnectionFactory, 256
- getByAddress
 - decaf::net::InetAddress, 1887, 1888
- getByte
 - activemq::commands::ActiveMQMapMessage, 320
 - activemq::util::PrimitiveList, 2791
 - activemq::util::PrimitiveMap, 2801
 - activemq::util::PrimitiveValueNode, 2824
 - cms::MapMessage, 2321
- getByteArray
 - activemq::util::PrimitiveList, 2791
 - activemq::util::PrimitiveMap, 2802
 - activemq::util::PrimitiveValueNode, 2824
 - decaf::internal::util::ByteArrayAdapter, 901
- getByteProperty
 - activemq::commands::ActiveMQMessageTemplate, 384
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2559
 - cms::Message, 2380
- getBytes
 - activemq::commands::ActiveMQMapMessage, 321
 - cms::MapMessage, 2321
- getBytesRead
 - decaf::util::zip::Deflater, 1599
 - decaf::util::zip::Inflater, 1896
- getBytesWritten
 - decaf::util::zip::Deflater, 1600
 - decaf::util::zip::Inflater, 1897
- getCacheSize
 - activemq::commands::WireFormatInfo, 3721
 - activemq::wireformat::openwire::OpenWireFormat, 2704
- getCapacity

- decaf::internal::util::ByteArrayAdapter, 901
- getCause
 - activemq::commands::BrokerError, 794
 - cms::CMSException, 1076
 - decaf::lang::Exception, 1716
 - decaf::lang::Throwable, 3539
- getChar
 - activemq::commands::ActiveMQMapMessage, 321
 - activemq::util::PrimitiveList, 2792
 - activemq::util::PrimitiveMap, 2802
 - activemq::util::PrimitiveValueNode, 2825
 - cms::MapMessage, 2321
 - decaf::internal::nio::ByteBuffer, 932, 933
 - decaf::internal::util::ByteArrayAdapter, 901
 - decaf::nio::ByteBuffer, 965, 966
- getCharArray
 - decaf::internal::util::ByteArrayAdapter, 902
- getCharCapacity
 - decaf::internal::util::ByteArrayAdapter, 902
- getChecksum
 - decaf::util::zip::CheckedInputStream, 1057
 - decaf::util::zip::CheckedOutputStream, 1059
- getCipherSuites
 - decaf::net::ssl::SSLParameters, 3328
- getClientID
 - activemq::core::ActiveMQConnection, 241
 - cms::Connection, 1170
- getClientId
 - activemq::commands::ActiveMQDestination, 283
 - activemq::commands::ConnectionInfo, 1260
 - activemq::commands::JournalTopicAck, 2042, 2043
 - activemq::commands::RemoveSubscriptionInfo, 3017
 - activemq::commands::SubscriptionInfo, 3435, 3436
 - activemq::core::ActiveMQConnectionFactory, 256
- getCloseTimeout
 - activemq::core::ActiveMQConnection, 241
 - activemq::core::ActiveMQConnectionFactory, 256
- getCluster
 - activemq::commands::Message, 2364
- getCMSCorrelationID
 - activemq::commands::ActiveMQMessageTemplate, 385
 - cms::Message, 2381
- getCMSDeliveryMode
 - activemq::commands::ActiveMQMessageTemplate, 385
 - cms::Message, 2381
- getCMSDestination
 - activemq::commands::ActiveMQDestination, 283
 - activemq::commands::ActiveMQMessageTemplate, 385
 - activemq::commands::ActiveMQQueue, 437
 - activemq::commands::ActiveMQTempQueue, 551
 - activemq::commands::ActiveMQTempTopic, 579
 - activemq::commands::ActiveMQTopic, 635
 - cms::Message, 2381
- getCMSExpiration
 - activemq::commands::ActiveMQMessageTemplate, 385
 - cms::Message, 2382
- getCMSMajorVersion
 - activemq::core::ActiveMQConnectionMetaData, 263
 - cms::ConnectionMetaData, 1288
- getCMSMessageID
 - activemq::commands::ActiveMQMessageTemplate, 386
 - cms::Message, 2382
- getCMSMinorVersion
 - activemq::core::ActiveMQConnectionMetaData, 263
 - cms::ConnectionMetaData, 1289
- getCMSPriority
 - activemq::commands::ActiveMQMessageTemplate, 386
 - cms::Message, 2383
- getCMSProperties
 - activemq::commands::ActiveMQQueue, 437
 - activemq::commands::ActiveMQTempQueue, 551
 - activemq::commands::ActiveMQTempTopic, 579
 - activemq::commands::ActiveMQTopic, 635
 - cms::Destination, 1612
- getCMSProviderName
 - activemq::core::ActiveMQConnectionMetaData, 263
 - cms::ConnectionMetaData, 1289
- getCMSRedelivered

- activemq::commands::ActiveMQMessageTemplate, 386
- cms::Message, 2383
- getCMSReplyTo
 - activemq::commands::ActiveMQMessageTemplate, 387
 - cms::Message, 2383
- getCMSTimestamp
 - activemq::commands::ActiveMQMessageTemplate, 387
 - cms::Message, 2384
- getCMSType
 - activemq::commands::ActiveMQMessageTemplate, 387
 - cms::Message, 2384
- getCMSVersion
 - activemq::core::ActiveMQConnectionMetaData, 264
 - cms::ConnectionMetaData, 1289
- getCMSXPropertyNames
 - activemq::core::ActiveMQConnectionMetaData, 264
 - cms::ConnectionMetaData, 1289
- getCollisionAvoidancePercent
 - activemq::core::policies::DefaultRedeliveryPolicy, 1571
 - activemq::core::RedeliveryPolicy, 2975
- getCommand
 - activemq::commands::ControlCommand, 1392
 - activemq::wireformat::stomp::StompFrame, 3400
- getCommandId
 - activemq::commands::BaseCommand, 697
 - activemq::commands::Command, 1108
 - activemq::commands::PartialCommand, 2729
- getCommands
 - activemq::state::TransactionState, 3624
- getComponents
 - activemq::util::CompositeData, 1131
- getConnectedBrokers
 - activemq::commands::ConnectionControl, 1174
- getConnection
 - activemq::commands::Message, 2364
 - activemq::core::ActiveMQSession, 476
- getConnectionFactory
 - activemq::cmsutil::CmsAccessor, 1070
- getConnectionId
 - activemq::commands::BrokerInfo, 827
 - activemq::commands::ConnectionError, 1202, 1203
- activemq::commands::ConnectionInfo, 1260
- activemq::commands::ConsumerId, 1333
- activemq::commands::DestinationInfo, 1616
- activemq::commands::LocalTransactionId, 2203
- activemq::commands::ProducerId, 2872
- activemq::commands::RemoveSubscriptionInfo, 3017
- activemq::commands::SessionId, 3164
- activemq::commands::TransactionInfo, 3596, 3597
- activemq::core::ActiveMQConnection, 241
- getConnectionInfo
 - activemq::core::ActiveMQConnection, 241
- getConsumerId
 - activemq::commands::ConsumerControl, 1304
 - activemq::commands::ConsumerInfo, 1361
 - activemq::commands::MessageAck, 2396
 - activemq::commands::MessageDispatch, 2429
 - activemq::commands::MessageDispatchNotification, 2464
 - activemq::commands::MessagePull, 2565, 2566
 - activemq::core::ActiveMQConsumer, 273
 - activemq::core::DispatchData, 1671
- getConsumerInfo
 - activemq::core::ActiveMQConsumer, 273
- getConsumerState
 - activemq::state::SessionState, 3219
- getConsumerStates
 - activemq::state::SessionState, 3219
- getContent
 - activemq::commands::Message, 2364, 2365
- getContext
 - decaf::internal::net::ssl::DefaultSSLContext, 1582
- getCorrelationId
 - activemq::commands::Message, 2365
 - activemq::commands::MessagePull, 2566
 - activemq::commands::Response, 3078
- getCount
 - decaf::util::concurrent::CountDownLatch, 1419
- getData
 - activemq::commands::DataArrayResponse, 1425
 - activemq::commands::DataResponse, 1478, 1479
 - activemq::commands::PartialCommand, 2730

- getDataStructure
 - activemq::commands::Message, 2365
- getDataStructureType
 - activemq::commands::ActiveMQBlobMessage, 168
 - activemq::commands::ActiveMQBytesMessage, 200
 - activemq::commands::ActiveMQDestination, 284
 - activemq::commands::ActiveMQMapMessage, 321
 - activemq::commands::ActiveMQMessage, 354
 - activemq::commands::ActiveMQObjectMessage, 398
 - activemq::commands::ActiveMQQueue, 437
 - activemq::commands::ActiveMQStreamMessage, 490
 - activemq::commands::ActiveMQTempDestination, 526
 - activemq::commands::ActiveMQTempQueue, 551
 - activemq::commands::ActiveMQTempTopic, 579
 - activemq::commands::ActiveMQTextMessage, 607
 - activemq::commands::ActiveMQTopic, 635
 - activemq::commands::BrokerError, 795
 - activemq::commands::BrokerId, 800
 - activemq::commands::BrokerInfo, 827
 - activemq::commands::ConnectionControl, 1174
 - activemq::commands::ConnectionError, 1203
 - activemq::commands::ConnectionId, 1233
 - activemq::commands::ConnectionInfo, 1260
 - activemq::commands::ConsumerControl, 1304
 - activemq::commands::ConsumerId, 1333
 - activemq::commands::ConsumerInfo, 1361
 - activemq::commands::ControlCommand, 1392
 - activemq::commands::DataArrayResponse, 1425
 - activemq::commands::DataResponse, 1479
 - activemq::commands::DataStructure, 1557
 - activemq::commands::DestinationInfo, 1616
 - activemq::commands::DiscoveryEvent, 1646
 - activemq::commands::ExceptionResponse, 1721
 - activemq::commands::FlushCommand, 1813
 - activemq::commands::IntegerResponse, 1957
 - activemq::commands::JournalQueueAck, 2016
 - activemq::commands::JournalTopicAck, 2043
 - activemq::commands::JournalTrace, 2071
 - activemq::commands::JournalTransaction, 2097
 - activemq::commands::KeepAliveInfo, 2124
 - activemq::commands::LastPartialCommand, 2157
 - activemq::commands::LocalTransactionId, 2203
 - activemq::commands::Message, 2365
 - activemq::commands::MessageAck, 2396
 - activemq::commands::MessageDispatch, 2429
 - activemq::commands::MessageDispatchNotification, 2464
 - activemq::commands::MessageId, 2497
 - activemq::commands::MessagePull, 2566
 - activemq::commands::NetworkBridgeFilter, 2615
 - activemq::commands::PartialCommand, 2730
 - activemq::commands::ProducerAck, 2841
 - activemq::commands::ProducerId, 2872
 - activemq::commands::ProducerInfo, 2900
 - activemq::commands::RemoveInfo, 2989
 - activemq::commands::RemoveSubscriptionInfo, 3017
 - activemq::commands::ReplayCommand, 3044
 - activemq::commands::Response, 3078
 - activemq::commands::SessionId, 3164
 - activemq::commands::SessionInfo, 3190
 - activemq::commands::ShutdownInfo, 3251
 - activemq::commands::SubscriptionInfo, 3436
 - activemq::commands::TransactionId, 3572
 - activemq::commands::TransactionInfo, 3597
 - activemq::commands::WireFormatInfo, 3721
 - activemq::commands::XATransactionId, 3767
 - activemq::wireformat::openwire::marshal::DataStreamMarshal, 1511
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM, 176

activemq::wireformat::openwire::marshal::v1::ActiveMQByteMessageMarshaller	2038
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMessageMarshaller	2067
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMessageMarshaller	2089
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller	2120
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMessageMarshaller	2146
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMessageMarshaller	2179
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	2225
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	2416
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	2455
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller	2484
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	2520
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	2585
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	2638
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2753
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	2864
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	2895
activemq::wireformat::openwire::marshal::v1::ConsumerGroupMarshaller	2911
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	3004
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	3020
activemq::wireformat::openwire::marshal::v1::ContentCommandMarshaller	3051
activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller	3104
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	3186
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	3201
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller	3261
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	3443
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3604
activemq::wireformat::openwire::marshal::v1::IntegrationResponseMarshaller	3745

activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	1820	openwire::marshal::v2::FlushCommandMarshaller	1820
3782			
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobWireMessageMarshaller	1964	openwire::marshal::v2::IntegerResponseMarshaller	1964
184			
activemq::wireformat::openwire::marshal::v2::ActiveMQByteWireMessageMarshaller	2022	openwire::marshal::v2::JournalQueueAckMarshaller	2022
231			
activemq::wireformat::openwire::marshal::v2::ActiveMQMapWireMessageMarshaller	2051	openwire::marshal::v2::JournalTopicAckMarshaller	2051
346			
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageWireMarshaller	2073	openwire::marshal::v2::JournalTraceMarshaller	2073
372			
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectWireMessageMarshaller	2104	openwire::marshal::v2::JournalTransactionMarshaller	2104
416			
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueWireMarshaller	2130	openwire::marshal::v2::KeepAliveInfoMarshaller	2130
458			
activemq::wireformat::openwire::marshal::v2::ActiveMQSequenceWireMessageMarshaller	2167	openwire::marshal::v2::LastPartialCommandMarshaller	2167
517			
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller	2209	openwire::marshal::v2::LocalTransactionMarshaller	2209
570			
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	2404	openwire::marshal::v2::MessageAckMarshaller	2404
598			
activemq::wireformat::openwire::marshal::v2::ActiveMQTextWireMessageMarshaller	2439	openwire::marshal::v2::MessageDispatchMarshaller	2439
630			
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicWireMarshaller	2472	openwire::marshal::v2::MessageDispatchMarshaller	2472
658			
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2500	openwire::marshal::v2::MessageIdMarshaller	2500
822			
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	2569	openwire::marshal::v2::MessagePullMarshaller	2569
853			
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2618	openwire::marshal::v2::NetworkBridgeMarshaller	2618
1198			
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2737	openwire::marshal::v2::PartialCommandMarshaller	2737
1205			
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	2844	openwire::marshal::v2::ProducerAckMarshaller	2844
1235			
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	2875	openwire::marshal::v2::ProducerIdMarshaller	2875
1265			
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller	2907	openwire::marshal::v2::ProducerInfoMarshaller	2907
1308			
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	2993	openwire::marshal::v2::RemoveInfoMarshaller	2993
1335			
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	3028	openwire::marshal::v2::RemoveSubscriptionMarshaller	3028
1367			
activemq::wireformat::openwire::marshal::v2::ContentCommandMarshaller	3055	openwire::marshal::v2::ReplayCommandMarshaller	3055
1395			
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	3090	openwire::marshal::v2::ResponseMarshaller	3090
1427			
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	3166	openwire::marshal::v2::SessionIdMarshaller	3166
1489			
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	3209	openwire::marshal::v2::SessionInfoMarshaller	3209
1619			
activemq::wireformat::openwire::marshal::v2::DiscardQueueMarshaller	3257	openwire::marshal::v2::ShutdownInfoMarshaller	3257
1652			
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	3459	openwire::marshal::v2::SubscriptionInfoMarshaller	3459
1728			

activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	1656
3620	
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	1732
3737	
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	1824
3774	
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller	1968
172	
activemq::wireformat::openwire::marshal::v3::ActiveMQBinaryMessageMarshaller	2030
211	
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	2055
330	
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	2077
356	
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	2108
400	
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMessageMarshaller	2134
442	
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	2163
501	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	2213
554	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	2408
582	
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	2443
610	
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	2476
638	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2512
802	
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	2577
833	
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2630
1178	
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2745
1209	
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2852
1239	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	2883
1269	
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2919
1312	
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	3000
1339	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	3024
1371	
activemq::wireformat::openwire::marshal::v3::ConnectionFactoryMarshaller	3059
1398	
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	3099
1431	
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	3182
1493	
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	3205
1623	

activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	1497
3269	
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	1627
3440	
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	1660
3608	
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	1740
3749	
activemq::wireformat::openwire::marshal::v3::XATransactionInfoMarshaller	1828
3786	
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobWireFormatMarshaller	1972
180	
activemq::wireformat::openwire::marshal::v4::ActiveMQByteWireFormatMarshaller	2034
219	
activemq::wireformat::openwire::marshal::v4::ActiveMQMapWireFormatMarshaller	2063
338	
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	2085
364	
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectWireFormatMarshaller	2116
408	
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueWireFormatMarshaller	2138
450	
activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceWireFormatMarshaller	2175
509	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	2221
562	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	2412
586	
activemq::wireformat::openwire::marshal::v4::ActiveMQTextWireFormatMarshaller	2451
614	
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	2480
642	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2504
806	
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	2581
837	
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	2634
1182	
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	2749
1213	
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	2848
1243	
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	2879
1273	
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	2903
1316	
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	3012
1343	
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	3040
1375	
activemq::wireformat::openwire::marshal::v4::ConnectionFactoryMarshaller	3047
1402	
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller	3086
1435	

activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	1410	activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller	1410
3170		activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller	1443
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	1443	activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller	1481
3213		activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller	1639
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	1481	activemq::wireformat::openwire::marshal::v5::DiscoveryEventManager	1668
3273		activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	1736
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	1639	activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	1836
3451		activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller	1980
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	1668	activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	2026
3616		activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller	2047
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	1736	activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller	2093
3741		activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller	2112
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	1836	activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller	2142
3778		activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	2171
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	1980	activemq::wireformat::openwire::marshal::v5::LocalTransactionMarshaller	2217
188		activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller	2420
activemq::wireformat::openwire::marshal::v5::ActiveMQByteMessageMarshaller	2026	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	2447
223		activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	2488
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	2047	activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller	2508
342		activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller	2573
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	2093	activemq::wireformat::openwire::marshal::v5::NetworkBridgeFactory	2626
368		activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	2741
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	2112	activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller	2856
412		activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	2887
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller	2142	activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	2915
454		activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller	3008
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	2171	activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionMarshaller	3036
513			
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	2217		
566			
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	2420		
594			
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	2447		
622			
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	2488		
650			
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	2508		
814			
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	2573		
845			
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	2626		
1190			
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	2741		
1221			
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	2856		
1251			
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	2887		
1281			
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	2915		
1324			
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	3008		
1351			
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	3036		
1383			

activemq::wireformat::openwire::marshal::v5::ReplyCommandMarshaller	openwire::marshal::v6::ConsumerIdMarsh
3067	1355
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	openwire::marshal::v6::ConsumerInfoMarsh
3095	1387
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	openwire::marshal::v6::ControlCommandMarsh
3178	1414
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	openwire::marshal::v6::DataArrayResponseMarsh
3197	1447
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	openwire::marshal::v6::DataResponseMarsh
3265	1485
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	openwire::marshal::v6::DestinationInfoMarsh
3447	1635
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	openwire::marshal::v6::DiscoveryEventManager
3600	1648
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	openwire::marshal::v6::ExceptionResponseMarsh
3729	1724
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	openwire::marshal::v6::FlushCommandMarsh
3790	1816
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobWireFormatMarshaller	openwire::marshal::v6::IntegerResponseMarsh
192	1960
activemq::wireformat::openwire::marshal::v6::ActiveMQByteWireFormatMarshaller	openwire::marshal::v6::JournalQueueAckMarsh
227	2018
activemq::wireformat::openwire::marshal::v6::ActiveMQMapWireFormatMarshaller	openwire::marshal::v6::JournalTopicAckMarsh
350	2059
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	openwire::marshal::v6::JournalTraceMarsh
376	2081
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectWireFormatMarshaller	openwire::marshal::v6::JournalTransactionMarsh
420	2100
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueWireFormatMarshaller	openwire::marshal::v6::KeepAliveInfoMarsh
462	2126
activemq::wireformat::openwire::marshal::v6::ActiveMQStreamWireFormatMarshaller	openwire::marshal::v6::LastPartialCommandMarsh
521	2160
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	openwire::marshal::v6::LocalTransactionMarsh
574	2205
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	openwire::marshal::v6::MessageAckMarsh
602	2401
activemq::wireformat::openwire::marshal::v6::ActiveMQTextWireFormatMarshaller	openwire::marshal::v6::MessageDispatchMarsh
626	2459
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller	openwire::marshal::v6::MessageDispatchMarsh
654	2468
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	openwire::marshal::v6::MessageIdMarsh
818	2516
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller	openwire::marshal::v6::MessagePullMarsh
849	2589
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller	openwire::marshal::v6::NetworkBridgeMarsh
1194	2622
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller	openwire::marshal::v6::PartialCommandMarsh
1225	2732
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller	openwire::marshal::v6::ProducerAckMarsh
1255	2860
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller	openwire::marshal::v6::ProducerIdMarsh
1285	2891
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller	openwire::marshal::v6::ProducerInfoMarsh
1328	2923

- activemq::wireformat::openwire::marshal::v6::RemoteInfoMarshaller; CmsTemplate, 1089
- 2996
- activemq::core::ActiveMQProducer, 425
- activemq::wireformat::openwire::marshal::v6::RemoteSubscriptionInfoMarshaller, 2552
- 3032
- getDeliverySequenceId
- activemq::wireformat::openwire::marshal::v6::ReplyCommandMarshaller; MessageDispatchNotification, 2464
- 3063
- activemq::wireformat::openwire::marshal::v6::ReplyDestinationMarshaller, 2464
- 3108
- activemq::cmsutil::CmsTemplate::ProducerExecutor, 2468
- activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 2468
- 3174
- activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2472
- activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 2972
- 3193
- activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3071
- activemq::wireformat::openwire::marshal::v6::ShutdownCommandMarshaller, 3071
- 3253
- activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3071
- activemq::wireformat::openwire::marshal::v6::SubscriberInfoMarshaller, 3071
- 3455
- activemq::commands::ConsumerControl, 1304
- activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 1304
- 3612
- activemq::commands::ConsumerInfo, 1361
- activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 1361
- 3733
- activemq::commands::DestinationInfo, 1616
- activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 1616
- 3770
- activemq::commands::JournalQueueAck, 2016
- getDefault
- decaf::net::ServerSocketFactory, 3148
- decaf::net::SocketFactory, 3304
- decaf::net::ssl::SSLContext, 3322
- decaf::net::ssl::SSLServerSocketFactory, 3336
- decaf::net::ssl::SSLSocketFactory, 3346
- getDefaultBacklog
- decaf::net::ServerSocket, 3142
- getDefaultCipherSuites
- decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1586
- 1586
- decaf::internal::net::ssl::DefaultSSLSocketFactory, 1592
- 1592
- decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2673
- 2673
- getDestinationResolver
- decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2695
- 2695
- decaf::net::ssl::SSLServerSocketFactory, 3336
- decaf::net::ssl::SSLSocketFactory, 3346
- getDefaultDestination
- activemq::cmsutil::CmsTemplate, 1088, 1089
- getDefaultDestinationName
- activemq::cmsutil::CmsTemplate, 1089
- getDefaultSSLParameters
- decaf::net::ssl::SSLContext, 3322
- getDelay
- decaf::util::concurrent::Delayed, 1609
- getDeliveryMode
- activemq::cmsutil::CachedProducer, 998
- activemq::commands::JournalTopicAck, 2043, 2044
- activemq::commands::Message, 2365, 2366
- activemq::commands::MessageAck, 2396, 2397
- activemq::commands::MessageDispatch, 2429
- activemq::commands::MessageDispatchNotification, 2465
- activemq::commands::MessagePull, 2566, 2567
- activemq::commands::ProducerInfo, 2900
- activemq::commands::SubscriptionInfo, 3436, 3437
- activemq::cmsutil::CmsDestinationAccessor, 1073
- getDestinationType
- activemq::commands::ActiveMQDestination, 284
- activemq::commands::ActiveMQQueue, 438
- activemq::commands::ActiveMQTempQueue, 552
- activemq::commands::ActiveMQTempTopic, 580
- activemq::commands::ActiveMQTopic, 636
- cms::Destination, 1612
- getDisableMessageID
- activemq::cmsutil::CachedProducer, 998
- activemq::core::ActiveMQProducer, 425

- cms::MessageProducer, 2552
- getDisableMessageTimeStamp
 - activemq::cmsutil::CachedProducer, 999
 - activemq::core::ActiveMQProducer, 426
 - cms::MessageProducer, 2552
- getDouble
 - activemq::commands::ActiveMQMapMessage, 322
 - activemq::util::PrimitiveList, 2792
 - activemq::util::PrimitiveMap, 2802
 - activemq::util::PrimitiveValueNode, 2825
 - cms::MapMessage, 2321
 - decaf::internal::nio::ByteBuffer, 933
 - decaf::internal::util::ByteArrayAdapter, 902
 - decaf::nio::ByteBuffer, 966
- getDoubleArray
 - decaf::internal::util::ByteArrayAdapter, 903
- getDoubleAt
 - decaf::internal::util::ByteArrayAdapter, 903
- getDoubleCapacity
 - decaf::internal::util::ByteArrayAdapter, 903
- getDoubleProperty
 - activemq::commands::ActiveMQMessageTemplate, 387
 - activemq::wireformat::openwire::utils::MessageProperty, 2560
 - cms::Message, 2384
- getDurableTopicPrefetch
 - activemq::core::policies::DefaultPrefetchPolicy, 1567
 - activemq::core::PrefetchPolicy, 2785
- getEnabledCipherSuites
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2662
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2666
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2679
 - decaf::net::ssl::SSLServerSocket, 3332
 - decaf::net::ssl::SSLSocket, 3340
- getEnabledProtocols
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2663
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2666
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2679
 - decaf::net::ssl::SSLServerSocket, 3332
 - decaf::net::ssl::SSLSocket, 3340
- getEncoded
 - decaf::security::auth::x500::X500Principal, 3762
 - decaf::security::cert::Certificate, 1008
 - decaf::security::Key, 2150
- getEnumeration
 - activemq::core::ActiveMQQueueBrowser, 440
 - cms::QueueBrowser, 2952
- getenv
 - decaf::lang::System, 3491, 3492
- getErrorCode
 - decaf::net::SocketError, 3298
- getErrorHandler
 - decaf::util::logging::Handler, 1853
- getErrorString
 - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2689
 - decaf::net::SocketError, 3298
- getException
 - activemq::commands::ConnectionError, 1203
 - activemq::commands::ExceptionResponse, 1722
- getExceptionClass
 - activemq::commands::BrokerError, 795
- getExceptionListener
 - activemq::core::ActiveMQConnection, 242
 - activemq::core::ActiveMQConnectionFactory, 250
 - activemq::core::ActiveMQSession, 476
 - cms::Connection, 1170
- getExpiration
 - activemq::commands::Message, 2366
- getFileDescriptor
 - decaf::net::SocketImpl, 3309
- getFilter
 - decaf::util::logging::Handler, 1853
 - decaf::util::logging::Logger, 2243
- getFirstMessageId
 - activemq::commands::MessageAck, 2397
- getFirstNakNumber
 - activemq::commands::ReplayCommand, 3045
- getFloat
 - activemq::commands::ActiveMQMapMessage, 322
 - activemq::util::PrimitiveList, 2792
 - activemq::util::PrimitiveMap, 2803
 - activemq::util::PrimitiveValueNode, 2825
 - cms::MapMessage, 2322
 - decaf::internal::nio::ByteBuffer, 934
 - decaf::internal::util::ByteArrayAdapter, 903
 - decaf::nio::ByteBuffer, 967

- getFloatArray
 - decaf::internal::util::ByteArrayAdapter, 904
- getFloatAt
 - decaf::internal::util::ByteArrayAdapter, 904
- getFloatCapacity
 - decaf::internal::util::ByteArrayAdapter, 904
- getFloatProperty
 - activemq::commands::ActiveMQMessageTemplateInfo, 388
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2560
 - cms::Message, 2385
- getFormat
 - decaf::security::Key, 2150
- getFormatId
 - activemq::commands::XATransactionId, 3767
- getFormatter
 - decaf::util::logging::Handler, 1853
- getFragment
 - activemq::util::CompositeData, 1131
 - decaf::internal::net::URIType, 3692
 - decaf::net::URI, 3665
- getFreeThreadCount
 - decaf::util::concurrent::ThreadPool, 3534
- getGlobalPool
 - decaf::internal::AprPool, 669
 - decaf::internal::DecafRuntime, 1564
- getGlobalTransactionId
 - activemq::commands::XATransactionId, 3768
- getGroupID
 - activemq::commands::Message, 2366
- getGroupSequence
 - activemq::commands::Message, 2366
- getHandlers
 - decaf::util::logging::Logger, 2244
- getHead
 - decaf::util::logging::Formatter, 1840
 - decaf::util::logging::XMLFormatter, 3794
- getHoldCount
 - decaf::util::concurrent::locks::ReentrantLock, 2979
- getHost
 - activemq::util::CompositeData, 1131
 - decaf::internal::net::URIType, 3693
 - decaf::net::URI, 3665
- getHostAddress
 - decaf::net::InetAddress, 1888
- getHostName
 - decaf::net::InetAddress, 1888
- getHostname
 - activemq::util::IdGenerator, 1862
- getId
 - activemq::state::TransactionState, 3624
 - decaf::lang::Thread, 3525
- getIndex
 - decaf::net::URISyntaxException, 3689
- getInetAddress
 - decaf::net::Socket, 3288
 - decaf::net::SocketImpl, 3310
- getInfo
 - activemq::state::ConnectionState, 1292
 - activemq::state::ConsumerState, 1390
 - activemq::state::ProducerState, 2926
 - activemq::state::SessionState, 3219
- getInitialDelayTime
 - activemq::transport::inactivity::InactivityMonitor, 1875
- getInitialReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1756
- getInitialRedeliveryDelay
 - activemq::core::policies::DefaultRedeliveryPolicy, 1571
 - activemq::core::RedeliveryPolicy, 2975
- getInput
 - decaf::net::URISyntaxException, 3690
- getInputStream
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2680
 - decaf::internal::net::tcp::TcpSocket, 3502
 - decaf::net::Socket, 3288
 - decaf::net::SocketImpl, 3310
- getInstance
 - activemq::transport::mock::MockTransport, 2595
 - activemq::transport::TransportRegistry, 3647
 - activemq::wireformat::WireFormatRegistry, 3753
 - decaf::util::concurrent::ThreadPool, 3534
 - decaf::util::logging::LogWriter, 2266
- getInt
 - activemq::commands::ActiveMQMapMessage, 322
 - activemq::util::PrimitiveList, 2793
 - activemq::util::PrimitiveMap, 2803
 - activemq::util::PrimitiveValueNode, 2825
 - cms::MapMessage, 2322
 - decaf::internal::nio::ByteBuffer, 934, 935
 - decaf::internal::util::ByteArrayAdapter, 905
 - decaf::nio::ByteBuffer, 967, 968

- getIntArray
 - decaf::internal::util::ByteArrayAdapter, 905
- getIntAt
 - decaf::internal::util::ByteArrayAdapter, 905
- getIntCapacity
 - decaf::internal::util::ByteArrayAdapter, 906
- getIntProperty
 - activemq::commands::ActiveMQMessageTemplate, 388
 - activemq::wireformat::openwire::utils::MessageProperty, 2560
 - cms::Message, 2385
- getIssuerUniqueID
 - decaf::security::cert::X509Certificate, 3764
- getIssuerX500Principal
 - decaf::security::cert::X509Certificate, 3764
- getKeepAlive
 - decaf::net::Socket, 3289
- getKey
 - decaf::util::Map::Entry, 1707
- getKeyUsage
 - decaf::security::cert::X509Certificate, 3764
- getLastDeliveredSequenceId
 - activemq::commands::RemoveInfo, 2990
 - activemq::core::ActiveMQConsumer, 274
 - activemq::core::ActiveMQSession, 477
- getLastMessageId
 - activemq::commands::MessageAck, 2397
- getLastNakNumber
 - activemq::commands::ReplayCommand, 3045
- getLastSequenceId
 - activemq::util::LongSequenceGenerator, 2302
- getLeastSignificantBits
 - decaf::util::UUID, 3709
- getLevel
 - decaf::util::logging::Handler, 1854
 - decaf::util::logging::Logger, 2244
 - decaf::util::logging::LogRecord, 2262
- getLimit
 - activemq::util::MemoryUsage, 2356
- getList
 - activemq::util::PrimitiveValueNode, 2826
- getLocalAddress
 - decaf::internal::net::tcp::TcpSocket, 3502
 - decaf::net::Socket, 3289
 - decaf::net::SocketImpl, 3310
- getLocalHost
 - decaf::net::InetAddress, 1888
- getLocalPort
 - decaf::net::ServerSocket, 3142
 - decaf::net::Socket, 3289
 - decaf::net::SocketImpl, 3310
- getLogger
 - decaf::util::logging::Logger, 2244
 - decaf::util::logging::LogManager, 2258
- getLoggerName
 - decaf::util::logging::LogRecord, 2262
- getLoggerNames
 - decaf::util::logging::LogManager, 2258
- getLogManager
 - decaf::util::logging::LogManager, 2258
- getMessageProperty
 - activemq::commands::ActiveMQMapMessage, 323
 - activemq::util::PrimitiveList, 2793
 - activemq::util::PrimitiveMap, 2804
 - activemq::util::PrimitiveValueNode, 2826
 - cms::MapMessage, 2322
 - decaf::internal::nio::ByteBuffer, 935
 - decaf::internal::util::ByteArrayAdapter, 906
 - decaf::nio::ByteBuffer, 968
- getLongArray
 - decaf::internal::util::ByteArrayAdapter, 906
- getLongAt
 - decaf::internal::util::ByteArrayAdapter, 906
- getLongCapacity
 - decaf::internal::util::ByteArrayAdapter, 907
- getLongProperty
 - activemq::commands::ActiveMQMessageTemplate, 389
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2560
 - cms::Message, 2385
- getMagic
 - activemq::commands::WireFormatInfo, 3722
- getManaged
 - decaf::internal::util::GenericResource, 1848
- getMap
 - activemq::commands::ActiveMQMapMessage, 323
 - activemq::util::PrimitiveValueNode, 2826
- getMapNames
 - activemq::commands::ActiveMQMapMessage, 323
 - cms::MapMessage, 2323
- getMarshaledForm
 - activemq::commands::BaseDataStructure, 766

- activemq::wireformat::MarshalAware, 2330
- getMarshaledProperties
 - activemq::commands::Message, 2366
 - activemq::commands::WireFormatInfo, 3722
- getMaxCacheSize
 - activemq::state::ConnectionStateTracker, 1296
 - activemq::transport::failover::FailoverTransport, 1756
- getMaximumPendingMessageLimit
 - activemq::commands::ConsumerInfo, 1362
- getMaximumRedeliveries
 - activemq::core::policies::DefaultRedeliveryPolicy, 1571
 - activemq::core::RedeliveryPolicy, 2975
- getMaxInactivityDuration
 - activemq::commands::WireFormatInfo, 3722
 - activemq::wireformat::openwire::OpenWireFormat, 2704
- getMaxInactivityDurationInitialDelay
 - activemq::commands::WireFormatInfo, 3722
- getMaxInactivityDurationInitialDelay
 - activemq::wireformat::openwire::OpenWireFormat, 2704
- getMaxPrefetchLimit
 - activemq::core::policies::DefaultPrefetchPolicy, 1567
 - activemq::core::PrefetchPolicy, 2785
- getMaxReconnectAttempts
 - activemq::transport::failover::FailoverTransport, 1756
- getMaxReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1756
- getMaxThreads
 - decaf::util::concurrent::ThreadPool, 3534
- getMessage
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2972
 - activemq::commands::BrokerError, 795
 - activemq::commands::JournalTrace, 2071
 - activemq::commands::MessageDispatch, 2429
 - activemq::core::DispatchData, 1671
 - cms::CMSException, 1076
 - decaf::lang::Exception, 1716
 - decaf::lang::Throwable, 3539
 - decaf::util::logging::LogRecord, 2263
- getMessageAck
 - activemq::commands::JournalQueueAck, 2016
- getMessageAvailableCount
 - activemq::core::ActiveMQConsumer, 274
- getMessageCount
 - activemq::commands::MessageAck, 2397
- getMessageId
 - activemq::commands::JournalTopicAck, 2044
 - activemq::commands::Message, 2366
 - activemq::commands::MessageDispatchNotification, 2465
 - activemq::commands::MessagePull, 2567
- getMessageListener
 - activemq::cmsutil::CachedConsumer, 994
 - activemq::core::ActiveMQConsumer, 274
 - cms::MessageConsumer, 2424
- getMessageProperties
 - activemq::commands::Message, 2366
- getMessageSelector
 - activemq::cmsutil::CachedConsumer, 994
 - activemq::core::ActiveMQConsumer, 274
 - activemq::core::ActiveMQQueueBrowser, 440
 - cms::MessageConsumer, 2424
 - cms::QueueBrowser, 2952
- getMessageSequenceId
 - activemq::commands::JournalTopicAck, 2044
- getMetaData
 - activemq::core::ActiveMQConnection, 242
 - cms::Connection, 1170
- getMimeType
 - activemq::commands::ActiveMQBlobMessage, 168
- getMostSignificantBits
 - decaf::util::UUID, 3709
- getName
 - activemq::commands::ActiveMQBlobMessage, 169
 - decaf::lang::Thread, 3525
 - decaf::security::auth::x500::X500Principal, 3762
 - decaf::security::Principal, 2831
 - decaf::util::logging::Level, 2188
 - decaf::util::logging::Logger, 2244
- getNeedClientAuth
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2663
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2667
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2680
 - decaf::net::ssl::SSLParameters, 3328
 - decaf::net::ssl::SSLServerSocket, 3333
 - decaf::net::ssl::SSLSocket, 3341

- getNetworkBrokerId
 - activemq::commands::NetworkBridgeFilter, 2616
- getNetworkConsumerPath
 - activemq::commands::ConsumerInfo, 1362
- getNetworkProperties
 - activemq::commands::BrokerInfo, 828
- getNetworkRuntime
 - decaf::internal::net::Network, 2613
- getNetworkTTL
 - activemq::commands::NetworkBridgeFilter, 2616
- getNextConsumerId
 - activemq::core::ActiveMQSession, 477
- getNextLocalTransactionId
 - activemq::core::ActiveMQConnection, 242
- getNextProducerId
 - activemq::core::ActiveMQSession, 477
- getNextSequenceId
 - activemq::util::LongSequenceGenerator, 2302
- getNextSessionId
 - activemq::core::ActiveMQConnection, 242
- getNextTempDestinationId
 - activemq::core::ActiveMQConnection, 243
- getNotAfter
 - decaf::security::cert::X509Certificate, 3764
- getNotBefore
 - decaf::security::cert::X509Certificate, 3764
- getNumReceivedMessageBeforeFail
 - activemq::transport::mock::MockTransport, 2596
- getNumReceivedMessages
 - activemq::transport::mock::MockTransport, 2596
- getNumSentKeepAlives
 - activemq::transport::mock::MockTransport, 2596
- getNumSentKeepAlivesBeforeFail
 - activemq::transport::mock::MockTransport, 2596
- getNumSentMessageBeforeFail
 - activemq::transport::mock::MockTransport, 2596
- getNumSentMessages
 - activemq::transport::mock::MockTransport, 2596
- getObjectId
 - activemq::commands::RemoveInfo, 2990
- getOOBInline
 - decaf::net::Socket, 3289
- getOperationType
 - activemq::commands::DestinationInfo, 1617
- getOption
 - decaf::internal::net::tcp::TcpSocket, 3503
 - decaf::net::SocketImpl, 3310
- getOptions
 - activemq::commands::ActiveMQDestination, 284
- getOrderedTarget
 - activemq::commands::ActiveMQDestination, 284
- getOriginalDestination
 - activemq::commands::Message, 2367
- getOriginalTransactionId
 - activemq::commands::Message, 2367
- getOutputStream
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2680
 - decaf::internal::net::tcp::TcpSocket, 3503
 - decaf::net::Socket, 3290
 - decaf::net::SocketImpl, 3311
- getParameters
 - activemq::util::CompositeData, 1131
- getParent
 - decaf::util::logging::Logger, 2245
- getParentId
 - activemq::commands::ConsumerId, 1333
 - activemq::commands::ProducerId, 2872
 - activemq::commands::SessionId, 3164
- getPassword
 - activemq::commands::ConnectionInfo, 1260, 1261
 - activemq::core::ActiveMQConnection, 243
 - activemq::core::ActiveMQConnectionFactory, 257
- getPath
 - activemq::util::CompositeData, 1131
 - decaf::internal::net::URIType, 3693
 - decaf::net::URI, 3666
- getPeerBrokerInfos
 - activemq::commands::BrokerInfo, 828
- getPhysicalName
 - activemq::commands::ActiveMQDestination, 284, 285
- getPooledThreadListener
 - decaf::util::concurrent::PooledThread, 2778
- getPoolSize
 - decaf::util::concurrent::ThreadPool, 3534
- getPort
 - decaf::internal::net::URIType, 3693
 - decaf::net::Socket, 3290
 - decaf::net::SocketImpl, 3311
 - decaf::net::URI, 3666
- getPreferredWireFormatInfo
 - activemq::wireformat::openwire::OpenWireFormat, 2705

- getPrefetch
 - activemq::commands::ConsumerControl, 1305
- getPrefetchPolicy
 - activemq::core::ActiveMQConnection, 243
 - activemq::core::ActiveMQConnectionFactory, 257
- getPrefetchSize
 - activemq::commands::ConsumerInfo, 1362
- getPreparedResult
 - activemq::state::TransactionState, 3624
- getPriority
 - activemq::cmsutil::CachedProducer, 999
 - activemq::cmsutil::CmsTemplate, 1089
 - activemq::commands::ConsumerInfo, 1362
 - activemq::commands::Message, 2367
 - activemq::core::ActiveMQProducer, 426
 - cms::MessageProducer, 2553
 - decaf::lang::Thread, 3525
- getProducerId
 - activemq::commands::Message, 2367
 - activemq::commands::MessageId, 2497, 2498
 - activemq::commands::ProducerAck, 2841
 - activemq::commands::ProducerInfo, 2900
 - activemq::core::ActiveMQProducer, 426
- getProducerInfo
 - activemq::core::ActiveMQProducer, 426
- getProducerSequenceId
 - activemq::commands::MessageId, 2498
- getProducerState
 - activemq::state::SessionState, 3219
- getProducerStates
 - activemq::state::SessionState, 3219
 - activemq::state::TransactionState, 3624
- getProducerWindowSize
 - activemq::core::ActiveMQConnection, 243
 - activemq::core::ActiveMQConnectionFactory, 257
- getProperties
 - activemq::commands::WireFormatInfo, 3722, 3723
 - activemq::util::ActiveMQProperties, 432, 433
 - activemq::wireformat::stomp::StompFrame, 3400, 3401
 - decaf::lang::System, 3492
 - decaf::util::logging::LogManager, 2259
- getProperty
 - activemq::util::ActiveMQProperties, 433
 - activemq::wireformat::stomp::StompFrame, 3401
 - cms::CMSProperties, 1080
 - decaf::lang::System, 3492
 - decaf::util::logging::LogManager, 2259
 - decaf::util::Properties, 2929, 2930
- getPropertyNames
 - activemq::commands::ActiveMQMessageTemplate, 389
 - cms::Message, 2386
- getProtocols
 - decaf::net::ssl::SSLParameters, 3328
- getProviderMajorVersion
 - activemq::core::ActiveMQConnectionMetaData, 264
 - cms::ConnectionMetaData, 1290
- getProviderMinorVersion
 - activemq::core::ActiveMQConnectionMetaData, 264
 - cms::ConnectionMetaData, 1290
- getProviderVersion
 - activemq::core::ActiveMQConnectionMetaData, 265
 - cms::ConnectionMetaData, 1290
- getPublicKey
 - decaf::security::cert::Certificate, 1008
- getQuery
 - decaf::internal::net::URIType, 3693
 - decaf::net::URI, 3666
- getQueue
 - activemq::core::ActiveMQQueueBrowser, 440
 - cms::QueueBrowser, 2952
- getQueueBrowserPrefetch
 - activemq::core::policies::DefaultPrefetchPolicy, 1567
 - activemq::core::PrefetchPolicy, 2786
- getQueueName
 - activemq::commands::ActiveMQQueue, 438
 - activemq::commands::ActiveMQTempQueue, 552
 - cms::Queue, 2947
 - cms::TemporaryQueue, 3517
- getQueuePrefetch
 - activemq::core::policies::DefaultPrefetchPolicy, 1568
 - activemq::core::PrefetchPolicy, 2786
- getRawAuthority
 - decaf::net::URI, 3666
- getRawFragment
 - decaf::net::URI, 3666
- getRawPath
 - decaf::net::URI, 3666
- getRawQuery
 - decaf::net::URI, 3667
- getRawSchemeSpecificPart
 - decaf::net::URI, 3667

- getRawUserInfo
 - decaf::net::URI, 3667
- getReadCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1876
- getReason
 - decaf::net::URISyntaxException, 3690
- getReceiveBufferSize
 - decaf::net::ServerSocket, 3142
 - decaf::net::Socket, 3290
- getReceiveTimeout
 - activemq::cmsutil::CmsTemplate, 1089
- getReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1756
- getReconnectTo
 - activemq::commands::ConnectionControl, 1174, 1175
- getRecoveringPullConsumers
 - activemq::state::ConnectionState, 1292
- getRedeliveryCounter
 - activemq::commands::Message, 2367
 - activemq::commands::MessageDispatch, 2429
- getRedeliveryDelay
 - activemq::core::policies::DefaultRedeliveryPolicy, 1571
 - activemq::core::RedeliveryPolicy, 2975
- getRedeliveryPolicy
 - activemq::core::ActiveMQConnection, 243
 - activemq::core::ActiveMQConnectionFactory, 257
 - activemq::core::ActiveMQConsumer, 274
- getRemaining
 - decaf::util::zip::Inflater, 1897
- getRemoteAddress
 - activemq::transport::failover::FailoverTransport, 1756
 - activemq::transport::IOTransport, 2008
 - activemq::transport::mock::MockTransport, 2596
 - activemq::transport::Transport, 3630
 - activemq::transport::TransportFilter, 3639
- getRemoteBlobUrl
 - activemq::commands::ActiveMQBlobMessage, 169
- getReplyTo
 - activemq::commands::Message, 2367
- getResourceLifecycleManager
 - activemq::cmsutil::CmsAccessor, 1070
 - activemq::cmsutil::SessionPool, 3217
- getResponse
 - activemq::transport::correlator::FutureResponse, 1844
- getResult
 - activemq::commands::IntegerResponse, 1958
- getReuseAddress
 - decaf::net::ServerSocket, 3143
 - decaf::net::Socket, 3290
- getRuntime
 - decaf::lang::Runtime, 3113
- getRuntimeLock
 - decaf::internal::net::Network, 2613
- getSafeValue
 - decaf::util::StlQueue, 3385
- getScheme
 - activemq::util::CompositeData, 1131
 - decaf::internal::net::URIType, 3693
 - decaf::net::URI, 3667
- getSchemeSpecificPart
 - decaf::internal::net::URIType, 3693
 - decaf::net::URI, 3667
- getSeedFromId
 - activemq::util::IdGenerator, 1862
- getSelector
 - activemq::commands::ConsumerInfo, 1362
 - activemq::commands::SubscriptionInfo, 3437
- getSendBufferSize
 - decaf::net::Socket, 3291
- getSendTimeout
 - activemq::core::ActiveMQConnection, 244
 - activemq::core::ActiveMQConnectionFactory, 257
 - activemq::core::ActiveMQProducer, 426
- getSequenceFromId
 - activemq::util::IdGenerator, 1862
- getServerSocketFactory
 - decaf::net::ssl::SSLContext, 3322
- getServiceName
 - activemq::commands::DiscoveryEvent, 1646
- getSession
 - activemq::cmsutil::PooledSession, 2775
- getSessionAcknowledgeMode
 - activemq::cmsutil::CmsAccessor, 1070
- getSessionId
 - activemq::commands::ConsumerId, 1333
 - activemq::commands::ProducerId, 2873
 - activemq::commands::SessionInfo, 3190, 3191
 - activemq::core::ActiveMQSession, 477
- getSessionInfo
 - activemq::core::ActiveMQSession, 477
- getSessionState
 - activemq::state::ConnectionState, 1292
- getSessionStates
 -

- activemq::state::ConnectionState, 1292
- getShort
 - activemq::commands::ActiveMQMapMessage, 323
 - activemq::util::PrimitiveList, 2794
 - activemq::util::PrimitiveMap, 2804
 - activemq::util::PrimitiveValueNode, 2826
 - cms::MapMessage, 2323
 - decaf::internal::nio::ByteBuffer, 936
 - decaf::internal::util::ByteArrayAdapter, 907
 - decaf::nio::ByteBuffer, 969
- getShortArray
 - decaf::internal::util::ByteArrayAdapter, 907
- getShortAt
 - decaf::internal::util::ByteArrayAdapter, 908
- getShortCapacity
 - decaf::internal::util::ByteArrayAdapter, 908
- getShortProperty
 - activemq::commands::ActiveMQMessageTemplate, 389
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2561
 - cms::Message, 2386
- getSigAlgName
 - decaf::security::cert::X509Certificate, 3764
- getSigAlgOID
 - decaf::security::cert::X509Certificate, 3764
- getSigAlgParams
 - decaf::security::cert::X509Certificate, 3764
- getSignature
 - decaf::security::cert::X509Certificate, 3764
- getSize
 - activemq::commands::ActiveMQTextMessage, 607
 - activemq::commands::Message, 2367
 - activemq::commands::ProducerAck, 2841
- getSocketFactory
 - decaf::net::ssl::SSLContext, 3323
- getSocketHandle
 - decaf::internal::net::tcp::TcpSocket, 3503
- getSoLinger
 - decaf::net::Socket, 3291
- getSoTimeout
 - decaf::net::ServerSocket, 3143
 - decaf::net::Socket, 3291
- getSource
 - decaf::internal::net::URIType, 3694
- getSourceFile
 - decaf::util::logging::LogRecord, 2263
- getSourceFunction
 - decaf::util::logging::LogRecord, 2263
- getSourceLine
 - decaf::util::logging::LogRecord, 2263
- getSSLParameters
 - decaf::net::ssl::SSLSocket, 3341
- getStackTrace
 - cms::CMSException, 1076
 - decaf::lang::Exception, 1716
 - decaf::lang::Throwable, 3539
- getStackTraceElements
 - activemq::commands::BrokerError, 795
- getStackTraceString
 - cms::CMSException, 1076
 - decaf::lang::Exception, 1717
 - decaf::lang::Throwable, 3539
- getStartupMaxReconnectAttempts
 - activemq::transport::failover::FailoverTransport, 1756
- getState
 - decaf::lang::Thread, 3525
- getString
 - activemq::commands::ActiveMQMapMessage, 324
 - activemq::util::PrimitiveList, 2794
 - activemq::util::PrimitiveMap, 2804
 - activemq::util::PrimitiveValueNode, 2827
 - cms::MapMessage, 2323
- getStringProperty
 - activemq::commands::ActiveMQMessageTemplate, 389
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2561
 - cms::Message, 2386
- getSubscriptionName
 - activemq::commands::RemoveSubscriptionInfo, 3017, 3018
 - activemq::commands::SubscriptionInfo, 3437
- getSubjectUniqueID
 - decaf::security::cert::X509Certificate, 3764
- getSubjectX500Principal
 - decaf::security::cert::X509Certificate, 3764
- getSubscribedDestination
 - activemq::commands::SubscriptionInfo, 3437
- getSubscriptionName
 - activemq::commands::ConsumerInfo, 1362
- getSubscriptionName
 - activemq::commands::JournalTopicAck, 2044
- getSupportedCipherSuites
 - decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1586

- decaf::internal::net::ssl::DefaultSSLSocketFactory, 1592
- decaf::internal::net::ssl::openssl::OpenSSLParameterTimeToLive, 2663
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2667
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2673
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2681
- decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2695
- decaf::net::ssl::SSLServerSocket, 3333
- decaf::net::ssl::SSLServerSocketFactory, 3336
- decaf::net::ssl::SSLSocket, 3341
- decaf::net::ssl::SSLSocketFactory, 3347
- getSupportedProtocols
 - decaf::internal::net::ssl::openssl::OpenSSLParameterTimeToLive, 2663
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2667
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2681
 - decaf::net::ssl::SSLServerSocket, 3333
 - decaf::net::ssl::SSLSocket, 3341
- getSupportedSSLParameters
 - decaf::net::ssl::SSLContext, 3323
- getTail
 - decaf::util::logging::Formatter, 1840
 - decaf::util::logging::XMLFormatter, 3794
- getTargetConsumerId
 - activemq::commands::Message, 2367, 2368
- getTBCertificate
 - decaf::security::cert::X509Certificate, 3764
- getTcpNoDelay
 - decaf::net::Socket, 3291
- getTempDestinations
 - activemq::state::ConnectionState, 1292
- getText
 - activemq::commands::ActiveMQTextMessage, 608
 - cms::TextMessage, 3519
- getThrown
 - decaf::util::logging::LogRecord, 2263
- getTime
 - decaf::util::Date, 1561
- getTimeout
 - activemq::commands::DestinationInfo, 1617
 - activemq::commands::MessagePull, 2567
 - activemq::transport::failover::FailoverTransport, 1756
- getTimestamp
 - decaf::util::logging::LogRecord, 2264
- activemq::commands::Message, 2368
- decaf::util::logging::LogRecord, 2263
- activemq::cmsutil::CachedProducer, 999
- activemq::cmsutil::CmsTemplate, 1089
- activemq::core::ActiveMQProducer, 427
- activemq::core::ActiveMQProducer, 2553
- getTopicName
 - activemq::commands::ActiveMQTempTopic, 580
 - activemq::commands::ActiveMQTopic, 636
 - cms::TemporaryTopic, 3518
 - cms::Topic, 3568
- getTopicPrefetch
 - activemq::core::policies::DefaultPrefetchPolicy, 1568
 - activemq::core::PrefetchPolicy, 2786
- getTrafficClass
 - decaf::net::Socket, 3292
- getTransactionContext
 - activemq::core::ActiveMQSession, 478
- getTransactionId
 - activemq::commands::JournalTopicAck, 2044
 - activemq::commands::JournalTransaction, 2097, 2098
 - activemq::commands::Message, 2368
 - activemq::commands::MessageAck, 2397
 - activemq::commands::TransactionInfo, 3597
 - activemq::core::ActiveMQTransactionContext, 662
- getTransactionState
 - activemq::state::ConnectionState, 1292
 - activemq::state::ProducerState, 2926
- getTransactionStates
 - activemq::state::ConnectionState, 1292
- getTransport
 - activemq::core::ActiveMQConnection, 244
 - activemq::transport::failover::BackupTransport, 690
- getTransportListener
 - activemq::transport::failover::FailoverTransport, 1756
 - activemq::transport::IOTransport, 2008
 - activemq::transport::mock::MockTransport, 2596
 - activemq::transport::Transport, 3630
 - activemq::transport::TransportFilter, 3639
- getTransportNames
 - activemq::transport::TransportRegistry, 3647
- getTreadId
 - decaf::util::logging::LogRecord, 2264

- getType
 - activemq::commands::JournalTransaction, 2098
 - activemq::commands::Message, 2368
 - activemq::commands::TransactionInfo, 3597
 - activemq::util::PrimitiveValueNode, 2827
 - decaf::security::cert::Certificate, 1008
- getUncaughtExceptionHandler
 - decaf::lang::Thread, 3525
- getUnconsumedMessages
 - activemq::core::ActiveMQSessionExecutor, 484
- getURI
 - activemq::transport::failover::URIPool, 3683
- getUri
 - activemq::transport::failover::BackupTransport, 691
- getUsage
 - activemq::util::MemoryUsage, 2356
- getClientMode
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2663
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2681
 - decaf::net::ssl::SSLSocket, 3342
- getParentHandlers
 - decaf::util::logging::Logger, 2245
- getUserID
 - activemq::commands::Message, 2368
- getUserInfo
 - decaf::internal::net::URIType, 3694
 - decaf::net::URI, 3668
- getUserName
 - activemq::commands::ConnectionInfo, 1261
- getUsername
 - activemq::core::ActiveMQConnection, 244
 - activemq::core::ActiveMQConnectionFactory, 258
- getValue
 - activemq::commands::BrokerId, 800
 - activemq::commands::ConnectionId, 1233
 - activemq::commands::ConsumerId, 1333
 - activemq::commands::LocalTransactionId, 2203
 - activemq::commands::ProducerId, 2873
 - activemq::commands::SessionId, 3164
 - activemq::util::PrimitiveValueNode, 2827
 - decaf::internal::net::SocketFileDescriptor, 3305
 - decaf::util::Map::Entry, 1707
 - decaf::util::zip::Adler32, 664
 - decaf::util::zip::Checksum, 1060
 - decaf::util::zip::CRC32, 1421
- getVersion
 - activemq::commands::WireFormatInfo, 3723
 - activemq::wireformat::openwire::OpenWireFormat, 2705
 - activemq::wireformat::stomp::StompWireFormat, 3408
 - activemq::wireformat::WireFormat, 3714
 - decaf::security::cert::X509Certificate, 3764
- getWantClientAuth
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2663
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2667
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2682
 - decaf::net::ssl::SSLParameters, 3328
 - decaf::net::ssl::SSLServerSocket, 3333
 - decaf::net::ssl::SSLSocket, 3342
- getWasPrepared
 - activemq::commands::JournalTransaction, 2098
- getWhen
 - decaf::util::TimerTask, 3555
- getWindowSize
 - activemq::commands::ProducerInfo, 2900
- getWireFormat
 - activemq::transport::mock::MockTransport, 2596
- getWireFormatNames
 - activemq::wireformat::WireFormatRegistry, 3754
- getWriteCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1876
- globalTransactionId
 - activemq::commands::XATransactionId, 3769
- good_match
 - internal_state, 1985
- groupID
 - activemq::commands::Message, 2374
- groupSequence
 - activemq::commands::Message, 2374
- GT_OFF
 - gzguts.h, 4204
- GUNZIP
 - inflate.h, 4205
- GZ_APPEND
 - gzguts.h, 4204
- gz_header
 - zlib.h, 4215

- gz_header_s, 1848
 - comm_max, 1849
 - comment, 1849
 - done, 1849
 - extra, 1849
 - extra_len, 1849
 - extra_max, 1849
 - hcrc, 1849
 - name, 1849
 - name_max, 1849
 - os, 1849
 - text, 1849
 - time, 1849
 - xflags, 1849
- gz_headerp
 - zlib.h, 4215
- GZ_NONE
 - gzguts.h, 4204
- GZ_READ
 - gzguts.h, 4204
- gz_state, 1849
 - direct, 1851
 - eof, 1851
 - err, 1851
 - fd, 1851
 - have, 1851
 - how, 1851
 - in, 1851
 - level, 1851
 - mode, 1851
 - msg, 1851
 - next, 1851
 - out, 1851
 - path, 1851
 - pos, 1851
 - raw, 1851
 - seek, 1851
 - size, 1851
 - skip, 1851
 - start, 1851
 - strategy, 1851
 - strm, 1851
 - want, 1851
- gz_statep
 - gzguts.h, 4204
- GZ_WRITE
 - gzguts.h, 4204
- GZBUFSIZE
 - gzguts.h, 4204
- gzFile
 - zlib.h, 4215
- gzguts.h
 - COPY, 4204
 - GT_OFF, 4204
 - GZ_APPEND, 4204
 - GZ_NONE, 4204
 - GZ_READ, 4204
 - gz_statep, 4204
 - GZ_WRITE, 4204
 - GZBUFSIZE, 4204
 - GZIP, 4204
 - local, 4204
 - LOOK, 4204
 - OF, 4204
 - ZLIB_INTERNAL, 4204
 - zstrerror, 4204
- gzhead
 - internal_state, 1985
- gzindex
 - internal_state, 1985
- GZIP
 - deflate.h, 4202
 - gzguts.h, 4204
- Handler
 - decaf::util::logging::Handler, 1853
- handleTransportFailure
 - activemq::transport::failover::FailoverTransport, 1757
- hasArray
 - decaf::internal::nio::ByteBuffer, 936
 - decaf::internal::nio::CharArrayBuffer, 1035
 - decaf::internal::nio::DoubleArrayBuffer, 1690
 - decaf::internal::nio::FloatArrayBuffer, 1798
 - decaf::internal::nio::IntArrayBuffer, 1929
 - decaf::internal::nio::LongArrayBuffer, 2289
 - decaf::internal::nio::ShortArrayBuffer, 3236
 - decaf::nio::ByteBuffer, 969
 - decaf::nio::CharBuffer, 1046
 - decaf::nio::DoubleBuffer, 1699
 - decaf::nio::FloatBuffer, 1807
 - decaf::nio::IntBuffer, 1937
 - decaf::nio::LongBuffer, 2297
 - decaf::nio::ShortBuffer, 3245
- hash_bits
 - internal_state, 1985
- hash_mask
 - internal_state, 1985
- hash_shift
 - internal_state, 1985
- hash_size
 - internal_state, 1985
- hashCode
 - decaf::security::auth::x500::X500Principal, 3762
- hasMoreMessages

- activemq::core::ActiveMQQueueBrowser, 440
- cms::MessageEnumeration, 2491
- hasMoreTokens
 - decaf::util::StringTokenizer, 3432
- hasNegotiator
 - activemq::wireformat::openwire::OpenWireFormat, 2705
 - activemq::wireformat::stomp::StompWireFormat, 3408
 - activemq::wireformat::WireFormat, 3714
- hasNext
 - decaf::util::Iterator, 2013
- hasPrevious
 - decaf::util::ListIterator, 2198
- hasProperty
 - activemq::util::ActiveMQProperties, 433
 - activemq::wireformat::stomp::StompFrame, 3401
 - cms::CMSProperties, 1081
 - decaf::util::Properties, 2930
- hasRemaining
 - decaf::nio::Buffer, 859
- hasUnconsumedMessages
 - activemq::core::ActiveMQSessionExecutor, 484
- have
 - gz_state, 1851
 - inflate_state, 1893
- HAVE_PTHREAD_H
 - activemq/util/Config.h, 3889
 - decaf/util/Config.h, 3890
- HAVE_UUID_T
 - activemq/util/Config.h, 3889
 - decaf/util/Config.h, 3890
- HAVE_UUID_UUID_H
 - activemq/util/Config.h, 3889
 - decaf/util/Config.h, 3890
- havedict
 - inflate_state, 1893
- HCRC
 - inflate.h, 4206
- hcrc
 - gz_header_s, 1849
- HCRC_STATE
 - deflate.h, 4202
- HEAD
 - inflate.h, 4205
- head
 - inflate_state, 1893
 - internal_state, 1985
- HEADER_ACK
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_CLIENT_ID
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_CONSUMERPRIORITY
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_CONTENTLENGTH
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_CORRELATIONID
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_DESTINATION
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_DISPATCH_ASYNC
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_EXCLUSIVE
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_EXPIRES
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_ID
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_JMSPRIORITY
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_LOGIN
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_MAXPENDINGMSGLIMIT
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_MESSAGE
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_MESSAGEID
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_NOLOCAL
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_OLDSUBSCRIPTIONNAME
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_PASSWORD
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- HEADER_PERSISTENT
 - activemq::wireformat::stomp::StompCommandConstants, 3397

HEADER_PREFETCHSIZE heap
 activemq::wireformat::stomp::StompCommandConstants, 3397
 internal_state, 1985
 heap_len
 HEADER_RECEIPT_REQUIRED internal_state, 1985
 activemq::wireformat::stomp::StompCommandConstants, 3397
 internal_state, 1985
 HEADER_RECEIPTID HEAP_SIZE
 activemq::wireformat::stomp::StompCommandConstants, 3397
 inflate, 4202
 HexStringParser
 HEADER_REDELIVERED decaf::internal::util::HexStringParser, 1856
 activemq::wireformat::stomp::StompCommandConstants, 3397
 activemq::wireformat::openwire::utils::HexTable, 1858
 HEADER_REDELIVERYCOUNT
 activemq::wireformat::stomp::StompCommandConstants, 3397
 internal_state, 1985
 HEADER_REPLYTO highestOneBit
 activemq::wireformat::stomp::StompCommandConstants, 3397
 decaf::lang::Integer, 1947
 decaf::lang::Long, 2273
 HEADER_REQUESTID hold
 activemq::wireformat::stomp::StompCommandConstants, 3397
 inflate, state, 1893
 hostname
 HEADER_RESPONSEID decaf::net::InetAddress, 1891
 activemq::wireformat::stomp::StompCommandConstants, 3397
 decaf::util::concurrent::TimeUnit, 3567
 HEADER_RETROACTIVE how
 activemq::wireformat::stomp::StompCommandConstants, 3397
 state, 1851
 HttpRetryException
 HEADER_SELECTOR decaf::net::HttpRetryException, 1859,
 activemq::wireformat::stomp::StompCommandConstants, 3397
 1860
 HUFFMAN_ONLY
 HEADER_SESSIONID decaf::util::zip::Deflater, 1604
 activemq::wireformat::stomp::StompCommandConstants, 3397
 ID_ACTIVEMQBLOBMESSAGE
 HEADER_SUBSCRIPTION activemq::commands::ActiveMQBlobMessage, 170
 activemq::wireformat::stomp::StompCommandConstants, 3397
 ID_ACTIVEMQBYTESMESSAGE
 HEADER_SUBSCRIPTIONNAME activemq::commands::ActiveMQBytesMessage, 169
 activemq::wireformat::stomp::StompCommandConstants, 3397
 ID_ACTIVEMQDESTINATION
 HEADER_TIMESTAMP activemq::commands::ActiveMQDestination, 189
 activemq::wireformat::stomp::StompCommandConstants, 3397
 ID_ACTIVEMQMAPMESSAGE
 HEADER_TRANSACTIONID activemq::commands::ActiveMQMapMessage, 188
 activemq::wireformat::stomp::StompCommandConstants, 3397
 ID_ACTIVEMQMESSAGE
 HEADER_TRANSFORMATION activemq::commands::ActiveMQMessage, 155
 activemq::wireformat::stomp::StompCommandConstants, 3397
 ID_ACTIVEMQOBJECTMESSAGE
 HEADER_TRANSFORMATION_ERROR activemq::commands::ActiveMQObjectMessage, 199
 activemq::wireformat::stomp::StompCommandConstants, 3397
 ID_ACTIVEMQQUEUE
 HEADER_TYPE activemq::commands::ActiveMQQueue, 138
 activemq::wireformat::stomp::StompCommandConstants, 3397
 ID_ACTIVEMQSTREAMMESSAGE

- activemq::commands::ActiveMQStreamMessage, 499
- activemq::commands::ExceptionResponse, 1722
- ID_ ACTIVEMQTEMPDESTINATION ID_ FLUSHCOMMAND
- activemq::commands::ActiveMQTempDestination, 526
- activemq::commands::FlushCommand, 1814
- ID_ ACTIVEMQTEMPQUEUE ID_ INTEGERRESPONSE
- activemq::commands::ActiveMQTempQueue, 552
- activemq::commands::IntegerResponse, 1958
- ID_ ACTIVEMQTEMPTOPIC ID_ JOURNALQUEUEACK
- activemq::commands::ActiveMQTempTopic, 580
- activemq::commands::JournalQueueAck, 2017
- ID_ ACTIVEMQTEXTMESSAGE ID_ JOURNALTOPICACK
- activemq::commands::ActiveMQTextMessage, 609
- activemq::commands::JournalTopicAck, 2045
- ID_ ACTIVEMQTOPIC ID_ JOURNALTRACE
- activemq::commands::ActiveMQTopic, 636
- activemq::commands::JournalTrace, 2072
- ID_ BROKERID ID_ JOURNALTRANSACTION
- activemq::commands::BrokerId, 801
- activemq::commands::JournalTransaction, 2098
- ID_ BROKERINFO ID_ KEEPALIVEINFO
- activemq::commands::BrokerInfo, 831
- activemq::commands::KeepAliveInfo, 2125
- ID_ CONNECTIONCONTROL ID_ LASTPARTIALCOMMAND
- activemq::commands::ConnectionControl, 1176
- activemq::commands::LastPartialCommand, 2158
- ID_ CONNECTIONERROR ID_ LOCALTRANSACTIONID
- activemq::commands::ConnectionError, 1204
- activemq::commands::LocalTransactionId, 2204
- ID_ CONNECTIONID ID_ MESSAGE
- activemq::commands::ConnectionId, 1234
- activemq::commands::Message, 2374
- ID_ CONNECTIONINFO ID_ MESSAGEACK
- activemq::commands::ConnectionInfo, 1263
- activemq::commands::MessageAck, 2399
- ID_ CONSUMERCONTROL ID_ MESSAGEDISPATCH
- activemq::commands::ConsumerControl, 1306
- activemq::commands::MessageDispatch, 2431
- ID_ CONSUMERID ID_ MESSAGEDISPATCHNOTIFICATION
- activemq::commands::ConsumerId, 1334
- activemq::commands::MessageDispatchNotification, 2466
- ID_ CONSUMERINFO ID_ MESSAGEID
- activemq::commands::ConsumerInfo, 1365
- activemq::commands::MessageId, 2499
- ID_ CONTROLCOMMAND ID_ MESSAGEPULL
- activemq::commands::ControlCommand, 1393
- activemq::commands::MessagePull, 2568
- ID_ DATAARRAYRESPONSE ID_ NETWORKBRIDGEFILTER
- activemq::commands::DataArrayResponse, 1425
- activemq::commands::NetworkBridgeFilter, 2616
- ID_ DATARESPONSE ID_ PARTIALCOMMAND
- activemq::commands::DataResponse, 1479
- activemq::commands::PartialCommand, 2731
- ID_ DESTINATIONINFO ID_ PRODUCERACK
- activemq::commands::DestinationInfo, 1618
- activemq::commands::ProducerAck, 2842
- ID_ DISCOVERYEVENT ID_ PRODUCERID
- activemq::commands::DiscoveryEvent, 1647
- activemq::commands::ProducerId, 2873
- ID_ EXCEPTIONRESPONSE ID_ PRODUCERINFO
- activemq::commands::ProducerInfo, 2902
- ID_ REMOVEINFO

- activemq::commands::RemoveInfo, 2991
- ID_REMOVESUBSCRIPTIONINFO
 - activemq::commands::RemoveSubscriptionInfo, 3019
- ID_REPLAYCOMMAND
 - activemq::commands::ReplayCommand, 3046
- ID_RESPONSE
 - activemq::commands::Response, 3079
- ID_SESSIONID
 - activemq::commands::SessionId, 3165
- ID_SESSIONINFO
 - activemq::commands::SessionInfo, 3191
- ID_SHUTDOWNINFO
 - activemq::commands::ShutdownInfo, 3252
- ID_SUBSCRIPTIONINFO
 - activemq::commands::SubscriptionInfo, 3438
- ID_TRANSACTIONID
 - activemq::commands::TransactionId, 3572
- ID_TRANSACTIONINFO
 - activemq::commands::TransactionInfo, 3598
- ID_WIREFORMATINFO
 - activemq::commands::WireFormatInfo, 3727
- ID_XATransactionID
 - activemq::commands::XATransactionId, 3769
- IdGenerator
 - activemq::util::IdGenerator, 1862
- IllegalArgumentException
 - decaf::lang::exceptions::IllegalArgumentException, 1864
- IllegalMonitorStateException
 - decaf::lang::exceptions::IllegalMonitorStateException, 1866, 1867
- IllegalStateException
 - cms::IllegalStateException, 1869
 - decaf::lang::exceptions::IllegalStateException, 1870, 1871
- IllegalThreadStateException
 - decaf::lang::exceptions::IllegalThreadStateException, 1872, 1873
- impl
 - decaf::net::Socket, 3297
- implAccept
 - decaf::net::ServerSocket, 3143
- in
 - decaf::io::FileDescriptor, 1769
 - gz_state, 1851
- InactivityMonitor
 - activemq::transport::inactivity::InactivityMonitor, 1875
- increaseUsage
 - activemq::util::MemoryUsage, 2356
 - activemq::util::Usage, 3702
- incrementAndGet
 - decaf::util::concurrent::atomic::AtomicInteger, 684
- indexOf
 - decaf::util::List, 2193
 - decaf::util::StlList, 3365
- IndexOutOfBoundsException
 - decaf::lang::exceptions::IndexOutOfBoundsException, 1878, 1879
- INDIVIDUAL_ACKNOWLEDGE
 - cms::Session, 3152
- Inet4Address
 - decaf::net::Inet4Address, 1881
- Inet6Address
 - decaf::net::Inet6Address, 1884
- InetAddress
 - decaf::net::Inet4Address, 1883
 - decaf::net::Inet6Address, 1884
 - decaf::net::InetAddress, 1887
- InetSocketAddress
 - decaf::net::InetSocketAddress, 1892
- inffast.h
 - OF, 4205
- inflate
 - decaf::util::zip::Inflater, 1897, 1898
- inflate.h
 - BAD, 4206
 - CHECK, 4206
 - CODELENS, 4206
 - COMMENT, 4206
 - COPY, 4206
 - COPY_, 4206
 - DICT, 4206
 - DICTID, 4206
 - DIST, 4206
 - DISTEXT, 4206
 - DONE, 4206
 - EXLEN, 4205
 - EXTRA, 4206
 - FLAGS, 4205
 - GUNZIP, 4205
 - HCRC, 4206
 - HEAD, 4205
 - inflate_mode, 4205
 - LEN, 4206
 - LEN_, 4206
 - LENEXT, 4206
 - LENGTH, 4206
 - LENLENS, 4206
 - LIT, 4206
 - MATCH, 4206

- MEM, 4206
- NAME, 4206
- OS, 4205
- STORED, 4206
- SYNC, 4206
- TABLE, 4206
- TIME, 4205
- TYPE, 4206
- TYPEDO, 4206
- inflate_mode
 - inflate.h, 4205
- inflate_state, 1892
 - back, 1893
 - bits, 1893
 - check, 1893
 - codes, 1893
 - distbits, 1893
 - distcode, 1893
 - dmax, 1893
 - extra, 1893
 - flags, 1893
 - have, 1893
 - havedict, 1893
 - head, 1893
 - hold, 1893
 - last, 1893
 - lenbits, 1893
 - lencode, 1893
 - length, 1893
 - lens, 1893
 - mode, 1893
 - ncode, 1893
 - ndist, 1893
 - next, 1893
 - nlen, 1893
 - offset, 1893
 - sane, 1893
 - total, 1893
 - was, 1893
 - wbits, 1893
 - whave, 1893
 - window, 1893
 - wnext, 1893
 - work, 1893
 - wrap, 1893
 - wsiz, 1893
- inflateBackInit
 - zlib.h, 4214
- inflateInit
 - zlib.h, 4214
- inflateInit2
 - zlib.h, 4215
- Inflater
 - decaf::util::zip::Inflater, 1896
- inflater
 - decaf::util::zip::InflaterInputStream, 1909
- InflaterInputStream
 - decaf::util::zip::InflaterInputStream, 1904, 1905
- INFO
 - decaf::util::logging::Level, 2190
- Info
 - decaf::util::logging, 140
- info
 - decaf::util::logging::Logger, 2245
 - decaf::util::logging::SimpleLogger, 3280
- inftrees.h
 - CODES, 4207
 - codetype, 4207
 - DISTS, 4207
 - ENOUGH, 4207
 - ENOUGH_DISTS, 4207
 - ENOUGH_LENS, 4207
 - LENS, 4207
 - OF, 4207
- INHERIT
 - decaf::util::logging::Level, 2190
- init
 - activemq::cmsutil::CmsAccessor, 1071
 - activemq::cmsutil::CmsDestinationAccessor, 1073
 - activemq::cmsutil::CmsTemplate, 1089
 - activemq::cmsutil::DestinationResolver, 1643
 - activemq::cmsutil::DynamicDestinationResolver, 1705
- INIT_STATE
 - deflate.h, 4202
- initCause
 - decaf::lang::Exception, 1717
 - decaf::lang::Throwable, 3540
- initializeLibrary
 - activemq::library::ActiveMQCPP, 278
- initializeNetworking
 - decaf::internal::net::Network, 2613
- initializeRuntime
 - decaf::lang::Runtime, 3113
- initSocketImpl
 - decaf::net::Socket, 3292
- inProgressClearRequired
 - activemq::core::ActiveMQConsumer, 275
- InputStream
 - decaf::io::InputStream, 1911
- inputStream
 - decaf::io::FilterInputStream, 1777
- InputStreamReader
 - decaf::io::InputStreamReader, 1920
- inReceive

- activemq::wireformat::openwire::OpenWireFormat
 - hash_shift, 1985
 - hash_size, 1985
 - 2705
- activemq::wireformat::stomp::StompWireFormat
 - head, 1985
 - 3409
 - heap, 1985
- activemq::wireformat::WireFormat, 3714
- ins_h
 - internal_state, 1985
- insert
 - decaf::internal::util::TimerTaskHeap, 3558
- IntArrayBuffer
 - decaf::internal::nio::IntArrayBuffer, 1925, 1926
- intBitsToFloat
 - decaf::lang::Float, 1785
- IntBuffer
 - decaf::nio::IntBuffer, 1933
- Integer
 - decaf::lang::Integer, 1944
- INTEGER_TYPE
 - activemq::util::PrimitiveValueNode, 2821
- IntegerResponse
 - activemq::commands::IntegerResponse, 1957
- IntegerResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 1976
 - activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 1964
 - activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 1968
 - activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 1972
 - activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 1980
 - activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 1960
- internal_state, 1982
 - bi_buf, 1985
 - bi_valid, 1985
 - bl_count, 1985
 - bl_desc, 1985
 - bl_tree, 1985
 - block_start, 1985
 - d_buf, 1985
 - d_desc, 1985
 - depth, 1985
 - dummy, 1985
 - dyn_dtree, 1985
 - dyn_ltree, 1985
 - good_match, 1985
 - gzhead, 1985
 - gzindex, 1985
 - hash_bits, 1985
 - hash_mask, 1985
 - hash_shift, 1985
 - hash_size, 1985
 - head, 1985
 - heap, 1985
 - heap_len, 1985
 - heap_max, 1985
 - high_water, 1985
 - ins_h, 1985
 - l_buf, 1985
 - l_desc, 1985
 - last_eob_len, 1985
 - last_flush, 1985
 - last_lit, 1985
 - level, 1985
 - lit_bufsize, 1985
 - lookahead, 1985
 - match_available, 1985
 - match_length, 1985
 - match_start, 1985
 - matches, 1985
 - max_chain_length, 1985
 - max_lazy_match, 1985
 - method, 1985
 - nice_match, 1985
 - pending, 1985
 - pending_buf_size, 1985
 - prev, 1985
 - prev_match, 1985
 - status, 1985
 - strm, 1985
 - strstart, 1985
 - w_bits, 1985
 - w_mask, 1985
 - w_size, 1985
 - window, 1985
 - window_size, 1985
 - wrap, 1985
- InternalCommandListener
 - activemq::transport::mock::InternalCommandListener, 1987
- InterruptedException
 - decaf::lang::exceptions::InterruptedException, 1988, 1989
- InterruptedIOException
 - decaf::io::InterruptedIOException, 1990, 1991
- intf
 - zconf.h, 4211

- int Value
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 1757
 - 2815
 - decaf::lang::Byte, 888
 - decaf::lang::Character, 1023
 - decaf::lang::Double, 1677
 - decaf::lang::Float, 1786
 - decaf::lang::Integer, 1947
 - decaf::lang::Long, 2273
 - decaf::lang::Number, 2654
 - decaf::lang::Short, 3224
 - decaf::util::concurrent::atomic::AtomicInteger, 684
 - decaf::util::logging::Level, 2188
- InvalidClientIdException
 - cms::InvalidClientIdException, 1993
- InvalidDestinationException
 - cms::InvalidDestinationException, 1994
- InvalidKeyException
 - decaf::security::InvalidKeyException, 1995, 1996
- InvalidMarkException
 - decaf::nio::InvalidMarkException, 1997, 1998
- InvalidSelectorException
 - cms::InvalidSelectorException, 2000
- InvalidStateException
 - decaf::lang::exceptions::InvalidStateException, 2001, 2002
- IOException
 - decaf::io::IOException, 2003, 2004
- IOTransport
 - activemq::transport::IOTransport, 2007
- IPos
 - deflate.h, 4202
- isAbsolute
 - decaf::internal::net::URIType, 3694
 - decaf::net::URI, 3668
- isAdvisory
 - activemq::commands::ActiveMQDestination, 285
- isAlive
 - decaf::lang::Thread, 3526
- isAlwaysSyncSend
 - activemq::core::ActiveMQConnection, 244
 - activemq::core::ActiveMQConnectionFactory, 258
- isAnyLocalAddress
 - decaf::net::Inet4Address, 1881
 - decaf::net::InetAddress, 1888
- isAutoAcknowledge
 - activemq::core::ActiveMQSession, 478
- isBackup
 - activemq::transport::failover::FailoverTransport, 1757
 - isBound
 - decaf::net::ServerSocket, 3143
 - decaf::net::Socket, 3292
 - isBrokerInfo
 - activemq::commands::BaseCommand, 697
 - activemq::commands::BrokerInfo, 828
 - activemq::commands::Command, 1108
 - isBrokerMasterConnector
 - activemq::commands::ConnectionInfo, 1261
 - isBrowser
 - activemq::commands::ConsumerInfo, 1362
 - isBusy
 - decaf::util::concurrent::PooledThread, 2778
 - isCacheEnabled
 - activemq::commands::WireFormatInfo, 3723
 - activemq::wireformat::openwire::OpenWireFormat, 2705
 - isCancelled
 - decaf::util::concurrent::Future, 1842
 - isClientAcknowledge
 - activemq::core::ActiveMQSession, 478
 - isClientMaster
 - activemq::commands::ConnectionInfo, 1261
 - isClose
 - activemq::commands::ConnectionControl, 1175
 - activemq::commands::ConsumerControl, 1305
 - isClosed
 - activemq::core::ActiveMQConnection, 244
 - activemq::core::ActiveMQConsumer, 275
 - activemq::core::ActiveMQProducer, 427
 - activemq::core::MessageDispatchChannel, 2434
 - activemq::transport::failover::BackupTransport, 691
 - activemq::transport::failover::FailoverTransport, 1757
 - activemq::transport::IOTransport, 2008
 - activemq::transport::mock::MockTransport, 2597
 - activemq::transport::tcp::TcpTransport, 3513
 - activemq::transport::Transport, 3631
 - activemq::transport::TransportFilter, 3639
 - decaf::internal::net::tcp::TcpSocket, 3503
 - decaf::io::FilterInputStream, 1774
 - decaf::io::FilterOutputStream, 1779
 - decaf::net::ServerSocket, 3143

- decaf::net::Socket, 3292
- isComposite
 - activemq::commands::ActiveMQDestination, 285
- isCompressed
 - activemq::commands::Message, 2368
- isConnected
 - activemq::transport::failover::FailoverTransport, 1757
 - activemq::transport::IOTransport, 2008
 - activemq::transport::mock::MockTransport, 2597
 - activemq::transport::tcp::TcpTransport, 3513
 - activemq::transport::Transport, 3631
 - activemq::transport::TransportFilter, 3639
 - decaf::internal::net::tcp::TcpSocket, 3504
 - decaf::net::Socket, 3292
- isConnectionAdvisory
 - activemq::commands::ActiveMQDestination, 285
- isConnectionInfo
 - activemq::commands::BaseCommand, 698
 - activemq::commands::Command, 1109
 - activemq::commands::ConnectionInfo, 1261
- isConnectionInterruptProcessingComplete
 - activemq::state::ConnectionState, 1293
- isConsumerAdvisory
 - activemq::commands::ActiveMQDestination, 285
- isConsumerInfo
 - activemq::commands::BaseCommand, 698
 - activemq::commands::Command, 1109
 - activemq::commands::ConsumerInfo, 1362
- isDeletedByBroker
 - activemq::commands::ActiveMQBlobMessage, 169
- isDigit
 - decaf::lang::Character, 1023
- isDispatchAsync
 - activemq::commands::ConsumerInfo, 1362
 - activemq::commands::ProducerInfo, 2900
 - activemq::core::ActiveMQConnection, 245
 - activemq::core::ActiveMQConnectionFactory, 258
- isDone
 - decaf::util::concurrent::Future, 1843
 - decaf::util::zip::DeflaterOutputStream, 1608
- isDroppable
 - activemq::commands::Message, 2368
- isDuplexConnection
 - activemq::commands::BrokerInfo, 828
- isDupsOkAcknowledge
 - activemq::core::ActiveMQSession, 478
- isEmpty
 - activemq::core::ActiveMQSessionExecutor, 484
- activemq::core::MessageDispatchChannel, 2434
- activemq::util::ActiveMQProperties, 433
- cms::CMSProperties, 1081
- decaf::internal::util::TimerTaskHeap, 3558
- decaf::lang::String, 3429
- decaf::util::AbstractCollection, 149
- decaf::util::Collection, 1103
- decaf::util::concurrent::ConcurrentStlMap, 1148
- decaf::util::concurrent::SynchronousQueue, 3482
- decaf::util::Map, 2312
- decaf::util::Properties, 2930
- decaf::util::StlList, 3366
- decaf::util::StlMap, 3376
- decaf::util::StlSet, 3394
- isEnabled
 - activemq::transport::failover::BackupTransportPool, 693
- isExclusive
 - activemq::commands::ActiveMQDestination, 285
 - activemq::commands::ConsumerInfo, 1363
 - activemq::commands::ConnectionControl, 1175
- isExpired
 - activemq::commands::Message, 2368
- isExplicitQosEnabled
 - activemq::cmsutil::CmsTemplate, 1090
- isFailOnClose
 - activemq::transport::mock::MockTransport, 2597
- isFailOnKeepAliveSends
 - activemq::transport::mock::MockTransport, 2597
- isFailOnReceiveMessage
 - activemq::transport::mock::MockTransport, 2597
- isFailOnSendMessage
 - activemq::transport::mock::MockTransport, 2597
- isFailOnStart
 - activemq::transport::mock::MockTransport, 2597
- isFailOnStop
 - activemq::transport::mock::MockTransport, 2597

- isFair
 - decaf::util::concurrent::locks::ReentrantLock, 2980
 - decaf::util::concurrent::Semaphore, 3131
- isFaultTolerant
 - activemq::commands::ConnectionControl, 1175
 - activemq::commands::ConnectionInfo, 1261
 - activemq::transport::failover::FailoverTransport, 1757
 - activemq::transport::IOTransport, 2008
 - activemq::transport::mock::MockTransport, 2597
 - activemq::transport::tcp::TcpTransport, 3513
 - activemq::transport::Transport, 3631
 - activemq::transport::TransportFilter, 3639
- isFaultTolerantConfiguration
 - activemq::commands::BrokerInfo, 829
- isFlush
 - activemq::commands::ConsumerControl, 1305
- isFull
 - activemq::util::MemoryUsage, 2357
 - activemq::util::Usage, 3702
- isHeldByCurrentThread
 - decaf::util::concurrent::locks::ReentrantLock, 2980
- isIndividualAcknowledge
 - activemq::core::ActiveMQSession, 478
- isInfinite
 - decaf::lang::Double, 1678
 - decaf::lang::Float, 1786
- isInitialized
 - activemq::transport::failover::FailoverTransport, 1757
- isInputShutdown
 - decaf::net::Socket, 3293
- isInTransaction
 - activemq::core::ActiveMQTransactionContext, 662
- isISOControl
 - decaf::lang::Character, 1024
- isKeepAliveInfo
 - activemq::commands::BaseCommand, 698
 - activemq::commands::Command, 1109
 - activemq::commands::KeepAliveInfo, 2124
- isKeepAliveResponseRequired
 - activemq::transport::inactivity::InactivityMonitor, 1876
- isLetter
 - decaf::lang::Character, 1024
- isLetterOrDigit
 - decaf::lang::Character, 1024
- isLinkLocalAddress
 - decaf::net::Inet4Address, 1881
 - decaf::net::InetAddress, 1889
- isLocked
 - decaf::util::concurrent::Lock, 2229
 - decaf::util::concurrent::locks::ReentrantLock, 2980
- isLoggable
 - decaf::util::logging::Filter, 1770
 - decaf::util::logging::Handler, 1854
 - decaf::util::logging::Logger, 2245
 - decaf::util::logging::StreamHandler, 3414
- isLoopbackAddress
 - decaf::net::Inet4Address, 1881
 - decaf::net::InetAddress, 1889
- isLowerCase
 - decaf::lang::Character, 1024
- isManageable
 - activemq::commands::ConnectionInfo, 1262
- isMarshalAware
 - activemq::commands::ActiveMQMapMessage, 324
 - activemq::commands::BaseDataStructure, 766
 - activemq::commands::Message, 2368
 - activemq::commands::WireFormatInfo, 3723
 - activemq::wireformat::MarshalAware, 2330
- isMasterBroker
 - activemq::commands::BrokerInfo, 829
- isMCGlobal
 - decaf::net::Inet4Address, 1882
 - decaf::net::InetAddress, 1889
- isMCLinkLocal
 - decaf::net::Inet4Address, 1882
 - decaf::net::InetAddress, 1889
- isMCNodeLocal
 - decaf::net::Inet4Address, 1882
 - decaf::net::InetAddress, 1889
- isMCOrgLocal
 - decaf::net::Inet4Address, 1882
 - decaf::net::InetAddress, 1890
- isMCSiteLocal
 - decaf::net::Inet4Address, 1882
 - decaf::net::InetAddress, 1890
- isMessage
 - activemq::commands::BaseCommand, 698
 - activemq::commands::Command, 1109
 - activemq::commands::Message, 2369
- isMessageAck
 - activemq::commands::BaseCommand, 698
 - activemq::commands::Command, 1109

- activemq::commands::MessageAck, 2397
- isMessageDispatch
 - activemq::commands::BaseCommand, 698
 - activemq::commands::Command, 1109
 - activemq::commands::MessageDispatch, 2429
- isMessageDispatchNotification
 - activemq::commands::BaseCommand, 698
 - activemq::commands::Command, 1109
 - activemq::commands::MessageDispatchNotification, 2465
- isMessageIdEnabled
 - activemq::cmsutil::CmsTemplate, 1090
- isMessageTimestampEnabled
 - activemq::cmsutil::CmsTemplate, 1090
- isMulticastAddress
 - decaf::net::Inet4Address, 1883
 - decaf::net::InetAddress, 1890
- isNaN
 - decaf::lang::Double, 1678
 - decaf::lang::Float, 1786
- isNetworkConnection
 - activemq::commands::BrokerInfo, 829
- isNetworkSubscription
 - activemq::commands::ConsumerInfo, 1363
- isNoLocal
 - activemq::cmsutil::CmsTemplate, 1090
 - activemq::commands::ConsumerInfo, 1363
- isNoRangeAcks
 - activemq::commands::ConsumerInfo, 1363
- isOpaque
 - decaf::internal::net::URIType, 3694
 - decaf::net::URI, 3668
- isOptimizedAcknowledge
 - activemq::commands::ConsumerInfo, 1363
- isOrdered
 - activemq::commands::ActiveMQDestination, 286
- isOutputShutdown
 - decaf::net::Socket, 3293
- isPending
 - activemq::threads::CompositeTask, 1132
 - activemq::transport::failover::BackupTransportPool, 693
 - activemq::transport::failover::CloseTransportsTask, 1067
 - activemq::transport::failover::FailoverTransport, 1758
- isPersistent
 - activemq::commands::Message, 2369
- isPrepared
 - activemq::state::TransactionState, 3624
- isProducerAck
 - activemq::commands::BaseCommand, 699
- activemq::commands::Command, 1109
- activemq::commands::ProducerAck, 2841
- isProducerAdvisory
 - activemq::commands::ActiveMQDestination, 286
- isProducerInfo
 - activemq::commands::BaseCommand, 699
 - activemq::commands::Command, 1110
 - activemq::commands::ProducerInfo, 2900
- isPushSubDomain
 - activemq::cmsutil::CmsDestinationAccessor, 1073
- isQueue
 - activemq::commands::ActiveMQDestination, 286
- isRandomize
 - activemq::transport::failover::FailoverTransport, 1758
 - activemq::transport::failover::URIPool, 3683
- isReadOnly
 - decaf::internal::nio::ByteBuffer, 937
 - decaf::internal::nio::CharArrayBuffer, 1035
 - decaf::internal::nio::DoubleArrayBuffer, 1691
 - decaf::internal::nio::FloatArrayBuffer, 1798
 - decaf::internal::nio::IntArrayBuffer, 1929
 - decaf::internal::nio::LongArrayBuffer, 2289
 - decaf::internal::nio::ShortArrayBuffer, 3237
 - decaf::nio::Buffer, 859
 - decaf::nio::ByteBuffer, 970
- isReadOnlyBody
 - activemq::commands::Message, 2369
- isReadOnlyProperties
 - activemq::commands::Message, 2369
- isRebalanceConnection
 - activemq::commands::ConnectionControl, 1175
- isRecievedByDFBridge
 - activemq::commands::Message, 2369
- isRemoveInfo
 - activemq::commands::BaseCommand, 699
 - activemq::commands::Command, 1110
 - activemq::commands::RemoveInfo, 2990
- isRemoveSubscriptionInfo
 - activemq::commands::BaseCommand, 699
 - activemq::commands::Command, 1110
 - activemq::commands::RemoveSubscriptionInfo, 3018
- isResponse
 - activemq::commands::BaseCommand, 699
 - activemq::commands::Command, 1110
 - activemq::commands::Response, 3078

- isResponseRequired
 - activemq::commands::BaseCommand, 699
 - activemq::commands::Command, 1110
- isRestoreConsumers
 - activemq::state::ConnectionStateTracker, 1296
- isRestoreProducers
 - activemq::state::ConnectionStateTracker, 1296
- isRestoreSessions
 - activemq::state::ConnectionStateTracker, 1296
- isRestoreTransaction
 - activemq::state::ConnectionStateTracker, 1296
- isResume
 - activemq::commands::ConnectionControl, 1175
- isRetroactive
 - activemq::commands::ConsumerInfo, 1363
- isRunning
 - activemq::core::ActiveMQSessionExecutor, 484
 - activemq::core::MessageDispatchChannel, 2434
- isScheduled
 - decaf::util::TimerTask, 3555
- isServerAuthority
 - decaf::internal::net::URIType, 3694
- isShutdownInfo
 - activemq::commands::BaseCommand, 700
 - activemq::commands::Command, 1110
 - activemq::commands::ShutdownInfo, 3251
- isSiteLocalAddress
 - decaf::net::Inet4Address, 1883
 - decaf::net::InetAddress, 1890
- isSizePrefixDisabled
 - activemq::commands::WireFormatInfo, 3723
 - activemq::wireformat::openwire::OpenWireFormat, 2706
- isSlaveBroker
 - activemq::commands::BrokerInfo, 829
- isStackTraceEnabled
 - activemq::commands::WireFormatInfo, 3724
 - activemq::wireformat::openwire::OpenWireFormat, 2706
- isStart
 - activemq::commands::ConsumerControl, 1305
- isStarted
 - activemq::core::ActiveMQConnection, 245
 - activemq::core::ActiveMQSession, 478
- isStop
 - activemq::commands::ConsumerControl, 1305
- isSuspend
 - activemq::commands::ConnectionControl, 1175
- isSynchronizationRegistered
 - activemq::core::ActiveMQConsumer, 275
- isTcpNoDelayEnabled
 - activemq::commands::WireFormatInfo, 3724
 - activemq::wireformat::openwire::OpenWireFormat, 2706
- isTemporary
 - activemq::commands::ActiveMQDestination, 286
- isTightEncodingEnabled
 - activemq::commands::WireFormatInfo, 3724
 - activemq::wireformat::openwire::OpenWireFormat, 2706
- isTopic
 - activemq::commands::ActiveMQDestination, 286
- isTrackMessages
 - activemq::state::ConnectionStateTracker, 1296
 - activemq::transport::failover::FailoverTransport, 1758
- isTrackTransactionProducers
 - activemq::state::ConnectionStateTracker, 1296
 - activemq::transport::failover::FailoverTransport, 1758
- isTrackTransactions
 - activemq::state::ConnectionStateTracker, 1296
- isTransacted
 - activemq::cmsutil::PooledSession, 2775
 - activemq::core::ActiveMQSession, 478
 - cms::Session, 3159
- isTransactionInfo
 - activemq::commands::BaseCommand, 700
 - activemq::commands::Command, 1110
 - activemq::commands::TransactionInfo, 3597
- isTransportFailed
 - activemq::core::ActiveMQConnection, 245
- isUpperCase
 - decaf::lang::Character, 1024
- isUseAsyncSend
 - activemq::core::ActiveMQConnection, 245
 - activemq::core::ActiveMQConnectionFactory, 258

- isUseCollisionAvoidance
 - activemq::core::policies::DefaultRedeliveryPolicy, 1572
 - activemq::core::RedeliveryPolicy, 2976
- isUseCompression
 - activemq::core::ActiveMQConnection, 245
 - activemq::core::ActiveMQConnectionFactory, iterator 258
- isUseExponentialBackOff
 - activemq::core::policies::DefaultRedeliveryPolicy, 1572
 - activemq::core::RedeliveryPolicy, 2976
 - activemq::transport::failover::FailoverTransport, 1758
- isValid
 - activemq::commands::WireFormatInfo, 3724
 - decaf::internal::net::URIType, 3694
- isValidDomainName
 - decaf::internal::net::URIHelper, 3676
- isValidHexChar
 - decaf::internal::net::URIHelper, 3677
- isValidHost
 - decaf::internal::net::URIHelper, 3677
- isValidIP4Word
 - decaf::internal::net::URIHelper, 3677
- isValidIP6Address
 - decaf::internal::net::URIHelper, 3677
- isValidIPv4Address
 - decaf::internal::net::URIHelper, 3678
- isWaitingForResponse
 - activemq::state::Tracked, 3569
- isWhitespace
 - decaf::lang::Character, 1024
- isWildcard
 - activemq::commands::ActiveMQDestination, 286
- isWireFormatInfo
 - activemq::commands::BaseCommand, 700
 - activemq::commands::Command, 1111
 - activemq::commands::WireFormatInfo, 3724
- itemExists
 - activemq::commands::ActiveMQMapMessage, 324
 - cms::MapMessage, 2323
- iterate
 - activemq::core::ActiveMQConsumer, 275
 - activemq::core::ActiveMQSessionExecutor, 485
 - activemq::threads::CompositeTaskRunner, 1134
 - activemq::threads::Task, 3495
- activemq::transport::failover::BackupTransportPool, 694
- activemq::transport::failover::CloseTransportsTask, 1067
- activemq::transport::failover::FailoverTransport, 1758
- decaf::lang::Iterable, 2012
- decaf::util::concurrent::SynchronousQueue, 3482, 3483
- decaf::util::PriorityQueue, 2836
- decaf::util::StlList, 3366
- decaf::util::StlQueue, 3386
- decaf::util::StlSet, 3394
- join
 - decaf::lang::Thread, 3526
- JournalQueueAck
 - activemq::commands::JournalQueueAck, 2015
- JournalQueueAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalQueueAck, 2038
 - activemq::wireformat::openwire::marshal::v2::JournalQueueAck, 2022
 - activemq::wireformat::openwire::marshal::v3::JournalQueueAck, 2030
 - activemq::wireformat::openwire::marshal::v4::JournalQueueAck, 2034
 - activemq::wireformat::openwire::marshal::v5::JournalQueueAck, 2026
 - activemq::wireformat::openwire::marshal::v6::JournalQueueAck, 2018
- JournalTopicAck
 - activemq::commands::JournalTopicAck, 2042
- JournalTopicAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTopicAck, 2067
 - activemq::wireformat::openwire::marshal::v2::JournalTopicAck, 2051
 - activemq::wireformat::openwire::marshal::v3::JournalTopicAck, 2055
 - activemq::wireformat::openwire::marshal::v4::JournalTopicAck, 2063
 - activemq::wireformat::openwire::marshal::v5::JournalTopicAck, 2047
 - activemq::wireformat::openwire::marshal::v6::JournalTopicAck, 2059
- JournalTrace
 - activemq::commands::JournalTrace, 2070
- JournalTraceMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTraceMa, 2089

- activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2073
- activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 2077
- activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 2085
- activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 2093
- activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, 2081
- JournalTransaction
 - activemq::commands::JournalTransaction, 2096
- JournalTransactionMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2120
 - activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2104
 - activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 2108
 - activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 2116
 - activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 2112
 - activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, 2100
- KeepAliveInfo
 - activemq::commands::KeepAliveInfo, 2123
- KeepAliveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2146
 - activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 2130
 - activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 2134
 - activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 2138
 - activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 2142
 - activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2126
- KeyException
 - decaf::security::KeyException, 2151, 2152
- KeyManagementException
 - decaf::security::KeyManagementException, 2154, 2155
- keySet
 - decaf::util::concurrent::ConcurrentStlMap, 1148
 - decaf::util::Map, 2313
 - decaf::util::StlMap, 3377
- l_buf

- inflater.h, 4206
- length
 - decaf::internal::nio::CharArrayBuffer, 1037
 - decaf::lang::ArrayPointer, 673
 - decaf::lang::CharSequence, 1054
 - decaf::lang::String, 3429
 - decaf::nio::CharBuffer, 1047
 - decaf::util::zip::InflaterInputStream, 1909
 - inflate_state, 1893
- LENGTH_CODES
 - deflate.h, 4202
- LENLENS
 - inflater.h, 4206
- LENS
 - inftrees.h, 4207
- lens
 - inflate_state, 1893
- Less
 - decaf::util::comparators::Less, 2183
- Level
 - decaf::util::logging::Level, 2187
- level
 - gz_state, 1851
 - internal_state, 1985
- Levels
 - decaf::util::logging, 140
- limit
 - decaf::nio::Buffer, 859
- LineNumber
 - activemq::commands::BrokerError::StackTraceElement, 3351
- List
 - decaf::util::List, 2192
- LIST_TYPE
 - activemq::util::PrimitiveValueNode, 2821
- listen
 - decaf::internal::net::tcp::TcpSocket, 3504
 - decaf::net::SocketImpl, 3311
- listener
 - activemq::transport::TransportFilter, 3643
- listIterator
 - decaf::util::List, 2194, 2195
 - decaf::util::StlList, 3367
- listValue
 - activemq::util::PrimitiveValueNode::PrimitiveValueNode, 2815
- LIT
 - inflater.h, 4206
- lit_bufsize
 - internal_state, 1985
- LITERALS
 - deflate.h, 4202
- load
 - decaf::util::Properties, 2930, 2931
- local
 - gzguts.h, 4204
 - zutil.h, 4217
- localPort
 - decaf::net::SocketImpl, 3313
- LocalTransactionId
 - activemq::commands::LocalTransactionId, 2202
- LocalTransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2225
 - activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2209
 - activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2213
 - activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 2221
 - activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2217
 - activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2205
- Lock
 - decaf::util::concurrent::Lock, 2228
- lock
 - activemq::core::MessageDispatchChannel, 2434
 - decaf::internal::util::concurrent::MutexImpl, 2610
 - decaf::internal::util::concurrent::SynchronizableImpl, 3474
 - decaf::io::InputStream, 1913
 - decaf::io::OutputStream, 2721
 - decaf::util::AbstractCollection, 150
 - decaf::util::concurrent::ConcurrentStlMap, 1149
 - decaf::util::concurrent::Lock, 2229
 - decaf::util::concurrent::locks::Lock, 2231
 - decaf::util::concurrent::locks::ReentrantLock, 2980
 - decaf::util::concurrent::Mutex, 2605
 - decaf::util::concurrent::Synchronizable, 3463
 - decaf::util::StlMap, 3377
 - decaf::util::StlQueue, 3386
 - ValueCount
 - decaf::util::concurrent::MutexHandle, 2609
- lock_owner
 - decaf::util::concurrent::MutexHandle, 2609
- lockInterruptibly
 - decaf::util::concurrent::locks::Lock, 2231
 - decaf::util::concurrent::locks::ReentrantLock, 2981
- log
 - decaf::util::logging::Logger, 2246

- decaf::util::logging::LogWriter, 2266, 2267
- decaf::util::logging::SimpleLogger, 3280
- LOGDECAF_DEBUG
 - LoggerDefines.h, 4308
- LOGDECAF_DEBUG_1
 - LoggerDefines.h, 4308
- LOGDECAF_DECLARE
 - LoggerDefines.h, 4308
- LOGDECAF_DECLARE_LOCAL
 - LoggerDefines.h, 4309
- LOGDECAF_ERROR
 - LoggerDefines.h, 4309
- LOGDECAF_FATAL
 - LoggerDefines.h, 4309
- LOGDECAF_INFO
 - LoggerDefines.h, 4309
- LOGDECAF_INITIALIZE
 - LoggerDefines.h, 4309
- LOGDECAF_WARN
 - LoggerDefines.h, 4309
- Logger
 - decaf::util::logging::Logger, 2241
- LoggerDefines.h
 - LOGDECAF_DEBUG, 4308
 - LOGDECAF_DEBUG_1, 4308
 - LOGDECAF_DECLARE, 4308
 - LOGDECAF_DECLARE_LOCAL, 4309
 - LOGDECAF_ERROR, 4309
 - LOGDECAF_FATAL, 4309
 - LOGDECAF_INFO, 4309
 - LOGDECAF_INITIALIZE, 4309
 - LOGDECAF_WARN, 4309
- LoggerHierarchy
 - decaf::util::logging::LoggerHierarchy, 2249
- LoggingInputStream
 - activemq::io::LoggingInputStream, 2250
- LoggingOutputStream
 - activemq::io::LoggingOutputStream, 2251
- LoggingTransport
 - activemq::transport::logging::LoggingTransport, 2253
- LogManager
 - decaf::util::logging::LogManager, 2257
- LogRecord
 - decaf::util::logging::LogRecord, 2262
- LogWriter
 - decaf::util::logging::LogWriter, 2266
- Long
 - decaf::lang::Long, 2270
- LONG_TYPE
 - activemq::util::PrimitiveValueNode, 2821
- LongArrayBuffer
 - decaf::internal::nio::LongArrayBuffer, 2285, 2286
- longBitsToDouble
 - decaf::lang::Double, 1678
- LongBuffer
 - decaf::nio::LongBuffer, 2293
- LongSequenceGenerator
 - activemq::util::LongSequenceGenerator, 2302
- longValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2815
 - decaf::lang::Byte, 889
 - decaf::lang::Character, 1024
 - decaf::lang::Double, 1679
 - decaf::lang::Float, 1787
 - decaf::lang::Integer, 1947
 - decaf::lang::Long, 2273
 - decaf::lang::Number, 2654
 - decaf::lang::Short, 3225
 - decaf::util::concurrent::atomic::AtomicInteger, 684
- LOOK
 - gzguts.h, 4204
- lookahead
 - internal_state, 1985
- LOOPBACK
 - decaf::net::InetAddress, 1891
- loopbackBytes
 - decaf::net::InetAddress, 1891
- looseMarshal
 - activemq::wireformat::openwire::marshal::BaseDataStreamMarshal, 748
 - activemq::wireformat::openwire::marshal::DataStreamMarshal, 1517
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMarshal, 176
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMarshal, 215
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestination, 295
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMarshal, 334
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessage, 360
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObject, 404
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueue, 446
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStream, 505
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTemplate, 531
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTemp, 558

activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	2225	activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	2225
590		activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	2416
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	2416	activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	2455
618		activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	2484
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller	2455	activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	2520
646		activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	2541
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	2484	activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	2585
716		activemq::wireformat::openwire::marshal::v1::NetworkBridgeMarshaller	2638
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	2520	activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	2754
810		activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	2864
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	2541	activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	2895
841		activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	2911
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	2585	activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	3004
1186		activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	3021
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2638	activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	3051
1217		activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	3104
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	2754	activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	3186
1247		activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	3201
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	2864	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	3261
1277		activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	3444
activemq::wireformat::openwire::marshal::v1::ConsumerGroupMarshaller	2895	activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	3577
1320		activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	3604
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	2911	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	3745
1347		activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	3782
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	3004	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	184
1379		activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	231
activemq::wireformat::openwire::marshal::v1::ContactCommandMarshaller	3021	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	307
1406			
activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller	3051		
1439			
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	3104		
1501			
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	3186		
1631			
activemq::wireformat::openwire::marshal::v1::DiscardResponseMarshaller	3201		
1664			
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	3261		
1744			
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3444		
1832			
activemq::wireformat::openwire::marshal::v1::IntegrationResponseMarshaller	3577		
1976			
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3604		
2038			
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3745		
2067			
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3782		
2089			
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	184		
2120			
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	231		
2146			
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	307		
2180			

activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller	1964
346	
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	2022
372	
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	2051
416	
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMessageMarshaller	2073
458	
activemq::wireformat::openwire::marshal::v2::ActiveMQSequenceMessageMarshaller	2104
517	
activemq::wireformat::openwire::marshal::v2::ActiveMQSimpleMessageMarshaller	2130
542	
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMessageMarshaller	2168
570	
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMessageMarshaller	2209
598	
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	2404
630	
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMessageMarshaller	2439
658	
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	2472
736	
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2501
822	
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	2533
853	
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2570
1198	
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2618
1206	
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	2737
1235	
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	2844
1265	
activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	2875
1308	
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	2907
1336	
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	2993
1367	
activemq::wireformat::openwire::marshal::v2::ContentCommandMarshaller	3028
1395	
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	3055
1427	
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	3091
1489	
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	3167
1620	
activemq::wireformat::openwire::marshal::v2::DiscardRequestMarshaller	3209
1652	
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	3257
1728	
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	3459
1820	

activemq::wireformat::openwire::marshal::v3::RemoteSubscriptionInfoMarshaller	1182
3025	
activemq::wireformat::openwire::marshal::v3::ReplyCommandMarshaller	1213
3059	
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	1243
3100	
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	1273
3182	
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	1316
3205	
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	1343
3269	
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	1375
3440	
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	1402
3585	
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	1435
3608	
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	1497
3749	
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	1627
3786	
activemq::wireformat::openwire::marshal::v4::ActiveMQBlockMessageMarshaller	1660
180	
activemq::wireformat::openwire::marshal::v4::ActiveMQByteArrayMessageMarshaller	1740
219	
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	1828
299	
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	1972
338	
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	2034
364	
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2063
408	
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	2085
450	
activemq::wireformat::openwire::marshal::v4::ActiveMQStackingMessageMarshaller	2116
509	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	2138
535	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	2176
562	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	2221
586	
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	2412
615	
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	2451
642	
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	2480
709	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2504
806	
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	2537
837	

activemq::wireformat::openwire::marshal::v4::MessagePublishFormat	566
2581	
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaler	594
2634	
activemq::wireformat::openwire::marshal::v4::PartialCommandFormat	622
2749	
activemq::wireformat::openwire::marshal::v4::ProductKeyWireFormat	650
2848	
activemq::wireformat::openwire::marshal::v4::ProductIdMarshaler	722
2879	
activemq::wireformat::openwire::marshal::v4::ProductInfoWireFormat	814
2904	
activemq::wireformat::openwire::marshal::v4::RemoteInfoFormat	845
3012	
activemq::wireformat::openwire::marshal::v4::RemoteSubscriptionInfoMarshaler	1190
3040	
activemq::wireformat::openwire::marshal::v4::ReplaceCommandFormat	1221
3047	
activemq::wireformat::openwire::marshal::v4::ResponseWireFormat	1251
3086	
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaler	1281
3170	
activemq::wireformat::openwire::marshal::v4::SessionInfoFormat	1324
3213	
activemq::wireformat::openwire::marshal::v4::ShutdownInfoFormat	1351
3273	
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoFormat	1383
3451	
activemq::wireformat::openwire::marshal::v4::TransactionIdFormat	1410
3588	
activemq::wireformat::openwire::marshal::v4::TransactionInfoFormat	1443
3616	
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaler	1481
3741	
activemq::wireformat::openwire::marshal::v4::XATransactionFormat	1639
3778	
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobWireFormat	1668
188	
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaler	1736
223	
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationFormat	1836
303	
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaler	1980
342	
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageFormat	2026
368	
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectWireMessageMarshaler	2047
412	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueFormat	2093
454	
activemq::wireformat::openwire::marshal::v5::ActiveMQSequenceMessageMarshaler	2112
513	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaler	2142
539	

activemq::wireformat::openwire::marshal::v5::LastReceivedCommandMarshaller	311	activemq::wireformat::openwire::marshal::v6::ActiveMQDestination
2172		
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller	350	activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessage
2217		
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller	376	activemq::wireformat::openwire::marshal::v6::ActiveMQMessage
2420		
activemq::wireformat::openwire::marshal::v5::MessageDispatchInfoMarshaller	420	activemq::wireformat::openwire::marshal::v6::ActiveMQObject
2447		
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	462	activemq::wireformat::openwire::marshal::v6::ActiveMQQueue
2488		
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller	521	activemq::wireformat::openwire::marshal::v6::ActiveMQStream
2508		
activemq::wireformat::openwire::marshal::v5::MessageMarshaller	546	activemq::wireformat::openwire::marshal::v6::ActiveMQTemplate
2524		
activemq::wireformat::openwire::marshal::v5::MessagePublishInfoMarshaller	574	activemq::wireformat::openwire::marshal::v6::ActiveMQTemp
2573		
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller	602	activemq::wireformat::openwire::marshal::v6::ActiveMQTemp
2626		
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	626	activemq::wireformat::openwire::marshal::v6::ActiveMQTextM
2741		
activemq::wireformat::openwire::marshal::v5::ProducerAckInfoMarshaller	654	activemq::wireformat::openwire::marshal::v6::ActiveMQTopic
2856		
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	729	activemq::wireformat::openwire::marshal::v6::BaseCommandM
2887		
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	818	activemq::wireformat::openwire::marshal::v6::BrokerIdMarsha
2915		
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller	849	activemq::wireformat::openwire::marshal::v6::BrokerInfoMarsh
3008		
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	1194	activemq::wireformat::openwire::marshal::v6::ConnectionCont
3036		
activemq::wireformat::openwire::marshal::v5::ReplyCommandMarshaller	1225	activemq::wireformat::openwire::marshal::v6::ConnectionError
3067		
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	1255	activemq::wireformat::openwire::marshal::v6::ConnectionIdMa
3095		
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	1285	activemq::wireformat::openwire::marshal::v6::ConnectionInfoM
3178		
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	1328	activemq::wireformat::openwire::marshal::v6::ConsumerContro
3197		
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	1355	activemq::wireformat::openwire::marshal::v6::ConsumerIdMar
3265		
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	1387	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMa
3447		
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller	1414	activemq::wireformat::openwire::marshal::v6::ControlComman
3574		
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	1447	activemq::wireformat::openwire::marshal::v6::DataArrayRespo
3600		
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	1485	activemq::wireformat::openwire::marshal::v6::DataResponseM
3729		
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	1635	activemq::wireformat::openwire::marshal::v6::DestinationInfoM
3790		
activemq::wireformat::openwire::marshal::v6::ActiveMQBlockWireFormatMarshaller	1648	activemq::wireformat::openwire::marshal::v6::DiscoveryEventM
192		
activemq::wireformat::openwire::marshal::v6::ActiveMQByteWireFormatMarshaller	1724	activemq::wireformat::openwire::marshal::v6::ExceptionRespo
227		

activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, openwire::marshal::v6::SubscriptionInfo
 1816 3455
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, openwire::marshal::v6::TransactionIdMa
 1960 3592
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, openwire::marshal::v6::TransactionInfo
 2019 3612
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, openwire::marshal::v6::WireFormatInfo
 2059 3733
 activemq::wireformat::openwire::marshal::v6::JournalTransWireFormat, openwire::marshal::v6::XATransactionId
 2081 3771
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller,
 2100 looseMarshalBrokerError
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller,
 2127 looseMarshalCachedObject
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller,
 2160 activemq::wireformat::openwire::marshal::v6::BaseDataStreamMa
 748
 activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller,
 2205 looseMarshalLong
 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller,
 2401 activemq::wireformat::openwire::marshal::v6::BaseDataStreamMa
 749
 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller,
 2459 activemq::wireformat::openwire::marshal::v6::BaseDataStreamMa
 749
 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller,
 2468 activemq::wireformat::openwire::OpenWireFormat,
 2706
 activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller,
 2516 looseMarshalObjectArray
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller,
 2545 activemq::wireformat::openwire::marshal::v6::BaseDataStreamMa
 749
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller,
 2589 activemq::wireformat::openwire::marshal::v6::BaseDataStreamMa
 750
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller,
 2622 looseUnmarshal
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller,
 2733 activemq::wireformat::openwire::marshal::v6::BaseDataStreamMa
 750
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller,
 2860 activemq::wireformat::openwire::marshal::v6::DataStreamMarshal
 1525
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller,
 2891 activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM
 176
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller,
 2923 activemq::wireformat::openwire::marshal::v1::ActiveMQBytes
 215
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller,
 2996 activemq::wireformat::openwire::marshal::v1::ActiveMQDestin
 295
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller,
 3032 activemq::wireformat::openwire::marshal::v1::ActiveMQMapM
 334
 activemq::wireformat::openwire::marshal::v6::ReplyCommandMarshaller, openwire::marshal::v1::ActiveMQMessa
 3063 361
 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, openwire::marshal::v1::ActiveMQObject
 3109 405
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, openwire::marshal::v1::ActiveMQQueue
 3174 447
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, openwire::marshal::v1::ActiveMQStream
 3193 505
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, openwire::marshal::v1::ActiveMQTemp
 3254 532

activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	2180
558	
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	2226
590	
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	2417
619	
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMessageMarshaller	2456
646	
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	2484
717	
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	2520
810	
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	2541
841	
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	2586
1186	
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2638
1218	
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	2754
1247	
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	2864
1277	
activemq::wireformat::openwire::marshal::v1::ConsumerGroupMarshaller	2895
1320	
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	2912
1347	
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	3005
1379	
activemq::wireformat::openwire::marshal::v1::ContactCommandMarshaller	3021
1407	
activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller	3052
1440	
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	3105
1502	
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	3186
1632	
activemq::wireformat::openwire::marshal::v1::DiscardResponseMarshaller	3201
1664	
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	3262
1745	
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3444
1832	
activemq::wireformat::openwire::marshal::v1::IntegrationResponseMarshaller	3578
1977	
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3604
2038	
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	3745
2067	
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	3783
2089	
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaller	184
2120	
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	231
2147	

activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller; openwire::marshal::v2::FlushCommand	1820
307	
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller; openwire::marshal::v2::IntegerResponse	1965
346	
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller; openwire::marshal::v2::JournalQueueAc	2023
373	
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller; openwire::marshal::v2::JournalTopicAck	2051
417	
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller; openwire::marshal::v2::JournalTraceMa	2074
459	
activemq::wireformat::openwire::marshal::v2::ActiveMQSequenceMessageMarshaller; openwire::marshal::v2::JournalTransact	2104
517	
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller; openwire::marshal::v2::KeepAliveInfoM	2131
543	
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller; openwire::marshal::v2::LastPartialComm	2168
570	
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller; openwire::marshal::v2::LocalTransaction	2210
598	
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller; openwire::marshal::v2::MessageAckMar	2405
631	
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller; openwire::marshal::v2::MessageDispatch	2440
658	
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller; openwire::marshal::v2::MessageDispatch	2472
737	
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller; openwire::marshal::v2::MessageIdMarsh	2501
822	
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller; openwire::marshal::v2::MessageMarshal	2533
853	
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller; openwire::marshal::v2::MessagePullMar	2570
1198	
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller; openwire::marshal::v2::NetworkBridgeF	2619
1206	
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller; openwire::marshal::v2::PartialCommand	2737
1236	
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller; openwire::marshal::v2::ProducerAckMar	2844
1265	
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller; openwire::marshal::v2::ProducerIdMars	2876
1308	
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller; openwire::marshal::v2::ProducerInfoMa	2908
1336	
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller; openwire::marshal::v2::RemoveInfoMars	2993
1368	
activemq::wireformat::openwire::marshal::v2::ContentCommandMarshaller; openwire::marshal::v2::RemoveSubscrip	3029
1395	
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller; openwire::marshal::v2::ReplayCommand	3056
1428	
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller; openwire::marshal::v2::ResponseMarsha	3091
1490	
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller; openwire::marshal::v2::SessionIdMarsha	3167
1620	
activemq::wireformat::openwire::marshal::v2::DiscardRequestMarshaller; openwire::marshal::v2::SessionInfoMars	3209
1653	
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller; openwire::marshal::v2::ShutdownInfoM	3258
1729	

activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	1371
3459	
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller	1399
3581	
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	1432
3620	
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	1494
3737	
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	1624
3775	
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobWireFormatMarshaller	1656
173	
activemq::wireformat::openwire::marshal::v3::ActiveMQByteWireFormatMarshaller	1733
212	
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	1824
291	
activemq::wireformat::openwire::marshal::v3::ActiveMQMapWireFormatMarshaller	1969
330	
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	2030
357	
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectWireFormatMarshaller	2055
401	
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueWireFormatMarshaller	2078
443	
activemq::wireformat::openwire::marshal::v3::ActiveMQStackWireFormatMarshaller	2108
501	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	2135
528	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	2164
554	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	2214
582	
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	2409
611	
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	2444
638	
activemq::wireformat::openwire::marshal::v3::BaseCommandWireFormatMarshaller	2476
703	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2512
803	
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	2529
833	
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2578
1179	
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2630
1210	
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2746
1240	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	2852
1269	
activemq::wireformat::openwire::marshal::v3::ConsumerGroupWireFormatMarshaller	2883
1312	
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	2920
1340	
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	

activemq::wireformat::openwire::marshal::v3::RemoteInfoMarshaller	3001	activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	837
activemq::wireformat::openwire::marshal::v3::RemoteSubscriptionInfoMarshaller	3025	activemq::wireformat::openwire::marshal::v4::ConnectionContainer	1182
activemq::wireformat::openwire::marshal::v3::ReplyCommandMarshaller	3060	activemq::wireformat::openwire::marshal::v4::ConnectionError	1214
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	3100	activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	1243
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	3182	activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1273
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	3205	activemq::wireformat::openwire::marshal::v4::ConsumerController	1316
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	3270	activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	1344
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	3440	activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1375
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	3585	activemq::wireformat::openwire::marshal::v4::ControlCommand	1403
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	3608	activemq::wireformat::openwire::marshal::v4::DataArrayResponse	1436
activemq::wireformat::openwire::marshal::v3::WireFormatMarshaller	3749	activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	1498
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	3787	activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	1628
activemq::wireformat::openwire::marshal::v4::ActiveMQBlockWireFormatMarshaller	180	activemq::wireformat::openwire::marshal::v4::DiscoveryEventManager	1660
activemq::wireformat::openwire::marshal::v4::ActiveMQByteMessageMarshaller	219	activemq::wireformat::openwire::marshal::v4::ExceptionResponse	1741
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	299	activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	1828
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	338	activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	1973
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	365	activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	2034
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	409	activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	2063
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	451	activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller	2085
activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceMessageMarshaller	509	activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	2116
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	535	activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	2139
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	562	activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	2176
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	586	activemq::wireformat::openwire::marshal::v4::LocalTransactionMarshaller	2222
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	615	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	2413
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	642	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2452
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	710	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2480
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	807	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	2505

activemq::wireformat::openwire::marshal::v4::MessageMarshaller	539
2537	
activemq::wireformat::openwire::marshal::v4::MessagePublishFormat	566
2582	
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller	594
2634	
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	623
2750	
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	650
2848	
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	723
2880	
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	814
2904	
activemq::wireformat::openwire::marshal::v4::RemoteInfoMarshaller	845
3013	
activemq::wireformat::openwire::marshal::v4::RemoteSubscriptionInfoMarshaller	1190
3041	
activemq::wireformat::openwire::marshal::v4::ReplaceCommandMarshaller	1222
3048	
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	1251
3087	
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	1281
3171	
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	1324
3213	
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	1351
3274	
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	1383
3452	
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	1411
3589	
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	1444
3616	
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	1482
3741	
activemq::wireformat::openwire::marshal::v4::XATransactionMarshaller	1640
3779	
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobWireFormatMarshaller	1668
188	
activemq::wireformat::openwire::marshal::v5::ActiveMQByteWireFormatMarshaller	1737
223	
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	1836
303	
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	1981
342	
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	2027
369	
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectWireMessageMarshaller	2047
413	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	2093
455	
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	2112
513	

activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshalleropenwire::marshal::v6::ActiveMQBytesS
 2143 227
 activemq::wireformat::openwire::marshal::v5::LastActiveCommandMarshalleropenwire::marshal::v6::ActiveMQDestin
 2172 311
 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshalleropenwire::marshal::v6::ActiveMQMapM
 2218 350
 activemq::wireformat::openwire::marshal::v5::MessageAckMarshalleropenwire::marshal::v6::ActiveMQMessa
 2421 377
 activemq::wireformat::openwire::marshal::v5::MessageDispatchInfoMarshalleropenwire::marshal::v6::ActiveMQObject
 2448 421
 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshalleropenwire::marshal::v6::ActiveMQQueue
 2488 463
 activemq::wireformat::openwire::marshal::v5::MessageIdMarshalleropenwire::marshal::v6::ActiveMQStream
 2509 521
 activemq::wireformat::openwire::marshal::v5::MessageMarshalleropenwire::marshal::v6::ActiveMQTemp
 2525 546
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshalleropenwire::marshal::v6::ActiveMQTemp
 2574 574
 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshalleropenwire::marshal::v6::ActiveMQTemp
 2626 602
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshalleropenwire::marshal::v6::ActiveMQTextM
 2742 627
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshalleropenwire::marshal::v6::ActiveMQTopicC
 2856 654
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshalleropenwire::marshal::v6::BaseCommandM
 2887 730
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshalleropenwire::marshal::v6::BrokerIdMarsh
 2916 818
 activemq::wireformat::openwire::marshal::v5::RemoteInfoMarshalleropenwire::marshal::v6::BrokerInfoMarsh
 3009 849
 activemq::wireformat::openwire::marshal::v5::RemoteSubscriptionInfoMarshalleropenwire::marshal::v6::ConnectionCont
 3037 1194
 activemq::wireformat::openwire::marshal::v5::ReplyCommandMarshalleropenwire::marshal::v6::ConnectionError
 3068 1226
 activemq::wireformat::openwire::marshal::v5::ResponseMarshalleropenwire::marshal::v6::ConnectionIdMa
 3096 1255
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshalleropenwire::marshal::v6::ConnectionInfoM
 3178 1285
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshalleropenwire::marshal::v6::ConsumerContro
 3197 1328
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshalleropenwire::marshal::v6::ConsumerIdMar
 3266 1355
 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshalleropenwire::marshal::v6::ConsumerInfoMa
 3448 1387
 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshalleropenwire::marshal::v6::ControlComman
 3574 1415
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshalleropenwire::marshal::v6::DataArrayRespo
 3600 1448
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshalleropenwire::marshal::v6::DataResponseM
 3730 1486
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshalleropenwire::marshal::v6::DestinationInfoM
 3791 1636
 activemq::wireformat::openwire::marshal::v6::ActiveMQBlockRefMsgMarshalleropenwire::marshal::v6::DiscoveryEventM
 192 1649

activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller,	3254
1725	
activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller,	3455
1816	
activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller,	3592
1961	
activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller,	3612
2019	
activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller,	3733
2059	
activemq::wireformat::openwire::marshal::v6::JournalTransactionInfoMarshaller,	3771
2081	
activemq::wireformat::openwire::marshal::v6::LossyUnmarshalByteBuffer,	
2101	activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,
activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller,	751
2127	looseUnmarshalByteArray
activemq::wireformat::openwire::marshal::v6::LastReceivedCommandMarshaller,	
2160	751
activemq::wireformat::openwire::marshal::v6::LossyUnmarshalByteBuffer,	
2206	activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,
activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller,	751
2401	looseUnmarshalConstByteArray
activemq::wireformat::openwire::marshal::v6::MessageDispatchInfoMarshaller,	
2460	752
activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller,	
2468	activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,
activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller,	752
2516	looseUnmarshalNestedObject
activemq::wireformat::openwire::marshal::v6::MessageMarshaller,	
2545	752
activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller,	
2590	2707
activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller,	
2622	activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,
activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller,	753
2733	lowestOneBit
activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller,	1948
2860	decaf::lang::Long, 2273
activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller,	
2891	MalformedURLException
activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller,	
2924	2303, 2304
activemq::wireformat::openwire::marshal::v6::RenamedInfoMarshaller,	
2997	activemq::commands::ConnectionInfo,
activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller,	2263
3033	Map
activemq::wireformat::openwire::marshal::v6::ReplyCommandMarshaller,	2307
3064	MAP_TYPE
activemq::wireformat::openwire::marshal::v6::ResponseMarshaller,	
3109	PrimitiveValueNode, 2821
activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller,	
3175	2815
activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller,	
3193	decaf::io::BufferedInputStream, 865

- decaf::io::ByteArrayInputStream, 948
- decaf::io::FilterInputStream, 1775
- decaf::io::InputStream, 1913
- decaf::io::PushbackInputStream, 2943
- decaf::io::Reader, 2962
- decaf::nio::Buffer, 860
- decaf::util::logging::SimpleLogger, 3281
- decaf::util::zip::InflaterInputStream, 1907
- Markblock
 - decaf::util::logging, 140
- MarkBlockLogger
 - decaf::util::logging::MarkBlockLogger, 2328
- markSupported
 - decaf::io::BufferedInputStream, 865
 - decaf::io::ByteArrayInputStream, 948
 - decaf::io::FilterInputStream, 1775
 - decaf::io::InputStream, 1913
 - decaf::io::PushbackInputStream, 2944
 - decaf::io::Reader, 2963
 - decaf::util::zip::InflaterInputStream, 1907
- marshal
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMapper, 2810
 - activemq::wireformat::openwire::OpenWireFormat, 2707
 - activemq::wireformat::openwire::utils::BooleanStream, 789
 - activemq::wireformat::stomp::StompWireFormat, 3409
 - activemq::wireformat::WireFormat, 3715
- marshalledProperties
 - activemq::commands::Message, 2374
- marshalledSize
 - activemq::wireformat::openwire::utils::BooleanStream, 789
- MarshallingSupport
 - activemq::util::MarshallingSupport, 2336
- marshalList
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMapper, 2810
- marshalMap
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMapper, 2811
- marshalPrimitive
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMapper, 2811
- marshalPrimitiveList
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMapper, 2811
- marshalPrimitiveMap
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMapper, 2812
- masterBroker
 - activemq::commands::BrokerInfo, 831
- MATCH
 - inflate.h, 4206
- match_available
 - internal_state, 1985
- match_length
 - internal_state, 1985
- match_start
 - internal_state, 1985
- matches
 - internal_state, 1985
- Math
 - decaf::lang::Math, 2341
- max
 - decaf::lang::Math, 2344, 2345
- MAX_BITS
 - deflate.h, 4202
- max_chain_length
 - internal_state, 1985
- max_code
 - tree_desc_s, 3648
- MAX_DIST
 - deflate.h, 4202
- max_insert_length
 - deflate.h, 4202
- max_lazy_match
 - internal_state, 1985
- MAX_MATCH
 - zutil.h, 4217
- MAX_MEM_LEVEL
 - zconf.h, 4211
- MAX_PREFETCH_SIZE
 - activemq::core::policies::DefaultPrefetchPolicy, 1569
- MAX_PRIORITY
 - decaf::lang::Thread, 3529
- MAX_RADIX
 - decaf::lang::Character, 1026
- MAX_VALUE
 - decaf::lang::Character, 1026
 - decaf::lang::Double, 1682
 - decaf::lang::Integer, 1956
 - decaf::lang::Long, 2281
 - decaf::lang::Short, 3229
- MAX_WBITS
 - zconf.h, 4211
- maxWireFormatMessageLimit
 - activemq::commands::ConsumerInfo, 1365
- MEM
 - inflate.h, 4206
- MemoryUsage
 - activemq::util::MemoryUsage, 2355

- MESSAGE
- activemq::wireformat::stomp::StompCommandConstants, 3397
 - Message
 - activemq::commands::Message, 2362
 - message
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2972
 - activemq::commands::JournalTrace, 2072
 - activemq::commands::MessageDispatch, 2431
 - decaf::lang::Exception, 1718
 - MessageAck
 - activemq::commands::MessageAck, 2395
 - messageAck
 - activemq::commands::JournalQueueAck, 2017
 - MessageAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2416
 - activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2404
 - activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2408
 - activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2412
 - activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2420
 - activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2400
 - messageCount
 - activemq::commands::MessageAck, 2399
 - MessageDispatch
 - activemq::commands::MessageDispatch, 2428
 - MessageDispatchChannel
 - activemq::core::MessageDispatchChannel, 2433
 - MessageDispatchMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2455
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2439
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2443
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2451
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2447
 - activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2459
 - MessageDispatchNotification
 - activemq::commands::MessageDispatchNotification, 2463
 - MessageDispatchNotificationMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2484
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2472
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2476
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2480
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2488
 - activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2468
 - MessageEOFException
 - cms::MessageEOFException, 2493
 - MessageFormatException
 - cms::MessageFormatException, 2494
 - MessageId
 - activemq::commands::MessageId, 2496
 - messageId
 - activemq::commands::JournalTopicAck, 2045
 - MessageIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2461
 - activemq::commands::MessagePull, 2568
 - MessageIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2500
 - activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2512
 - activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2504
 - activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2508
 - activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2516
 - activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 2516
 - MessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2541
 - activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2537
 - activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2533
 - activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2529
 - activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2537
 - activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2531
 - MessageNotReadableException
 - cms::MessageNotReadableException, 2549

- MessageNotWriteableException
 - cms::MessageNotWriteableException, 2550
- MessagePropertyInterceptor
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2559
- MessagePull
 - activemq::commands::MessagePull, 2565
- MessagePullMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2585
 - activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2569
 - activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2577
 - activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2581
 - activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2573
 - activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2589
- messageSequenceId
 - activemq::commands::JournalTopicAck, 2045
- method
 - internal_state, 1985
- MethodName
 - activemq::commands::BrokerError::StackTraceElement, 3351
- MICROSECONDS
 - decaf::util::concurrent::TimeUnit, 3567
- MILLISECONDS
 - decaf::util::concurrent::TimeUnit, 3567
- min
 - decaf::lang::Math, 2346, 2347
- MIN_LOOKAHEAD
 - deflate.h, 4202
- MIN_MATCH
 - zutil.h, 4217
- MIN_PRIORITY
 - decaf::lang::Thread, 3529
- MIN_RADIX
 - decaf::lang::Character, 1026
- MIN_VALUE
 - decaf::lang::Byte, 893
 - decaf::lang::Character, 1026
 - decaf::lang::Double, 1682
 - decaf::lang::Float, 1790
 - decaf::lang::Integer, 1956
 - decaf::lang::Long, 2281
 - decaf::lang::Short, 3229
- MINUTES
 - decaf::util::concurrent::TimeUnit, 3567
- MockTransport
 - activemq::transport::mock::MockTransport, 2594
- mode
 - gzipHeader, 1851
 - inflate_state, 1893
- modifiedUtf8ToAscii
 - activemq::util::MarshallingSupport, 2336
- msg
 - gz_state, 1851
 - z_stream_s, 3795
- Mutex
 - decaf::util::concurrent::Mutex, 2605
- MutexPullMarshaller
 - decaf::util::AbstractCollection, 155
- mutex
 - decaf::util::concurrent::ConditionHandle, 1163
- MutexHandle
 - decaf::util::concurrent::MutexHandle, 2609
- NAME
 - inflate.h, 4206
- name
 - gz_header_s, 1849
 - name_max
 - gz_header_s, 1849
- NAME_STATE
 - deflate.h, 4202
- nameUUIDFromBytes
 - decaf::util::UUID, 3709
- NaN
 - decaf::lang::Double, 1682
 - decaf::lang::Float, 1790
- NANOSECONDS
 - decaf::util::concurrent::TimeUnit, 3567
- nanoTime
 - decaf::lang::System, 3493
- narrow
 - activemq::transport::failover::FailoverTransport, 1758
 - activemq::transport::IOTransport, 2008
 - activemq::transport::mock::MockTransport, 2598
 - activemq::transport::Transport, 3631
 - activemq::transport::TransportFilter, 3640
- ncode
 - inflate_state, 1893
- ndist
 - inflate_state, 1893
- needsDictionary
 - decaf::util::zip::Inflater, 1898
- needsInput
 - decaf::util::zip::Deflater, 1600
 - decaf::util::zip::Inflater, 1899

- NEGATIVE_INFINITY
 - decaf::lang::Double, 1683
 - decaf::lang::Float, 1791
- Network
 - decaf::internal::net::Network, 2612
- NetworkBridgeFilter
 - activemq::commands::NetworkBridgeFilter, 2615
- NetworkBridgeFilterMarshaller
 - activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2638
 - activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2618
 - activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2630
 - activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2634
 - activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2626
 - activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2622
- networkBrokerId
 - activemq::commands::NetworkBridgeFilter, 2616
- networkConnection
 - activemq::commands::BrokerInfo, 831
- networkConsumerPath
 - activemq::commands::ConsumerInfo, 1365
- networkProperties
 - activemq::commands::BrokerInfo, 831
- networkSubscription
 - activemq::commands::ConsumerInfo, 1365
- networkTTL
 - activemq::commands::NetworkBridgeFilter, 2616
- NEW
 - decaf::lang::Thread, 3523
- newCondition
 - decaf::util::concurrent::locks::Lock, 2232
 - decaf::util::concurrent::locks::ReentrantLock, 2981
- newThread
 - decaf::util::concurrent::ThreadFactory, 3530
- next
 - activemq::transport::TransportFilter, 3643
 - decaf::security::SecureRandom, 3119
 - decaf::util::Iterator, 2013
 - decaf::util::Random, 2954
 - gz_state, 1851
 - inflate_state, 1893
- next_in
 - z_stream_s, 3795
- next_out
 - z_stream_s, 3795
- nextBoolean
 - decaf::util::Random, 2955
- nextBytes
 - decaf::security::SecureRandom, 3119, 3120
 - decaf::util::Random, 2955
- nextDouble
 - decaf::util::Random, 2955
- nextFloat
 - decaf::util::Random, 2955
- nextGaussian
 - decaf::util::Random, 2955
- nextIndex
 - decaf::util::ListIterator, 2199
- nextInt
 - decaf::util::Random, 2955
- nextLong
 - decaf::util::Random, 2955
- nextMessage
 - activemq::core::ActiveMQQueueBrowser, 440
 - cms::MessageEnumeration, 2491
- nextToken
 - decaf::util::StringTokenizer, 3432
- nice_match
 - internal_state, 1985
- nlen
 - inflate_state, 1893
- NO_COMPRESSION
 - decaf::util::zip::Deflater, 1604
- NO_MAXIMUM_REDELIVERIES
 - activemq::core::RedeliveryPolicy, 2977
- node
 - decaf::util::UUID, 3710
- noLocal
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2972
 - activemq::commands::ConsumerInfo, 1365
- NON_PERSISTENT
 - cms::DeliveryMode, 1610
- noRangeAcks
 - activemq::commands::ConsumerInfo, 1365
- NORM_PRIORITY
 - decaf::lang::Thread, 3529
- normalize
 - decaf::net::URI, 3668
- NoRouteToHostException
 - decaf::net::NoRouteToHostException, 2641, 2642
- NoSuchAlgorithmException
 - decaf::security::NoSuchAlgorithmException, 2644
- NoSuchElementException

- decaf::lang::exceptions::NoSuchElementException, 2646, 2647
- NoSuchProviderException
 - decaf::security::NoSuchProviderException, 2649
- notify
 - activemq::core::MessageDispatchChannel, 2434
 - decaf::internal::util::concurrent::ConditionImpl, 1164
 - decaf::internal::util::concurrent::SynchronizableImpl, 3474
 - decaf::io::InputStream, 1913
 - decaf::io::OutputStream, 2722
 - decaf::util::AbstractCollection, 150
 - decaf::util::concurrent::ConcurrentStlMap, 1149
 - decaf::util::concurrent::Mutex, 2605
 - decaf::util::concurrent::Synchronizable, 3464
 - decaf::util::StlMap, 3377
 - decaf::util::StlQueue, 3386
- notifyAll
 - activemq::core::MessageDispatchChannel, 2435
 - decaf::internal::util::concurrent::ConditionImpl, 1165
 - decaf::internal::util::concurrent::SynchronizableImpl, 3474
 - decaf::io::InputStream, 1914
 - decaf::io::OutputStream, 2722
 - decaf::util::AbstractCollection, 150
 - decaf::util::concurrent::ConcurrentStlMap, 1149
 - decaf::util::concurrent::Mutex, 2606
 - decaf::util::concurrent::Synchronizable, 3465
 - decaf::util::StlMap, 3378
 - decaf::util::StlQueue, 3386
- Null
 - decaf::util::logging, 140
- NULL_TYPE
 - activemq::util::PrimitiveValueNode, 2821
 - activemq::wireformat::openwire::OpenWireFormat, 2712
- NullPointerException
 - decaf::lang::exceptions::NullPointerException, 2651, 2652
- NUM_OPTIONS
 - activemq::core::ActiveMQConstants, 267
- NUM_PARAMS
 - activemq::core::ActiveMQConstants, 268
- NumberFormatException
 - decaf::lang::exceptions::NumberFormatException, 2656, 2657
- numberOfLeadingZeros
 - decaf::lang::Integer, 1948
 - decaf::lang::Long, 2274
- numberOfTrailingZeros
 - decaf::lang::Integer, 1948
 - decaf::lang::Long, 2274
- numWaiting
 - decaf::util::concurrent::ConditionHandle, 1163
- numWake
 - decaf::util::concurrent::ConditionHandle, 1163
- objectId
 - activemq::commands::RemoveInfo, 2991
- OF
 - deflate.h, 4202
 - gzguts.h, 4204
 - inffast.h, 4205
 - inftrees.h, 4207
 - zconf.h, 4211
 - zlib.h, 4215
 - zutil.h, 4217
- OFF
 - decaf::util::logging::Level, 2190
- on
 - decaf::util::logging, 140
- offer
 - decaf::util::concurrent::BlockingQueue, 778
 - decaf::util::concurrent::SynchronousQueue, 3483
 - decaf::util::PriorityQueue, 2836
 - decaf::util::Queue, 2949
- offset
 - decaf::internal::nio::CharArrayBuffer, 1037
 - inflate_state, 1893
- onCommand
 - activemq::core::ActiveMQConnection, 245
 - activemq::transport::correlator::ResponseCorrelator, 3082
 - activemq::transport::DefaultTransportListener, 1594
 - activemq::transport::failover::FailoverTransportListener, 1767
 - activemq::transport::inactivity::InactivityMonitor, 1876
 - activemq::transport::logging::LoggingTransport, 2253
 - activemq::transport::mock::InternalCommandListener, 1987
 - activemq::transport::TransportFilter, 3640

- activemq::transport::TransportListener, 3644
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2714
- oneway
 - activemq::core::ActiveMQConnection, 246
 - activemq::core::ActiveMQSession, 478
 - activemq::transport::correlator::ResponseCorrelator, 3083
 - activemq::transport::failover::FailoverTransport, 1759
 - activemq::transport::inactivity::InactivityMonitor, 1876
 - activemq::transport::IOTransport, 2009
 - activemq::transport::logging::LoggingTransport, 2253
 - activemq::transport::mock::MockTransport, 2598
 - activemq::transport::Transport, 3632
 - activemq::transport::TransportFilter, 3640
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2715
- onException
 - activemq::core::ActiveMQConnection, 246
 - activemq::transport::DefaultTransportListener, 1594
 - activemq::transport::failover::BackupTransport, 691
 - activemq::transport::failover::FailoverTransportListener, 1767
 - activemq::transport::inactivity::InactivityMonitor, 1876
 - activemq::transport::TransportFilter, 3641
 - activemq::transport::TransportListener, 3644
- cms::ExceptionListener, 1719
- onMessage
 - cms::MessageListener, 2523
- onProducerAck
 - activemq::core::ActiveMQProducer, 427
- onPropertiesReset
 - decaf::util::logging::PropertiesChangeListener, 2937
- onPropertyChanged
 - decaf::util::logging::PropertiesChangeListener, 2937
- onResponse
 - activemq::state::Tracked, 3569
- onSend
 - activemq::commands::ActiveMQBytesMessage, 200
 - activemq::commands::ActiveMQMessageTemplate, 390
- activemq::commands::ActiveMQStreamMessage, 490
- activemq::commands::Message, 2369
- onTaskComplete
 - decaf::util::concurrent::TaskListener, 3496
- onTaskCompleted
 - decaf::util::concurrent::PooledThreadListener, 2780
 - decaf::util::concurrent::ThreadPool, 3534
- onTaskException
 - decaf::util::concurrent::PooledThreadListener, 2780
 - decaf::util::concurrent::TaskListener, 3496
 - decaf::util::concurrent::ThreadPool, 3535
- onTaskStarted
 - decaf::util::concurrent::PooledThreadListener, 2780
 - decaf::util::concurrent::ThreadPool, 3535
- onTransportException
 - activemq::transport::correlator::ResponseCorrelator, 3083
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2715
- op
 - code, 1097
- opaque
 - z_stream_s, 3795
- OPEN_FAILURE
 - decaf::util::logging::ErrorManager, 1711
- OpenSSLContextSpi
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2660
- OpenSSLServerSocket
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2666
- OpenSSLServerSocketFactory
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2671
- OpenSSLSocket
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2661
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2678
- OpenSSLSocketException
 - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2687, 2688
- OpenSSLSocketFactory
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2661
 - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2692
- OpenSSLSocketInputStream
 - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2697

- OpenSSLSocketOutputStream
 - decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2699
- OpenWireFormat
 - activemq::wireformat::openwire::OpenWireFormat, 2703
- OpenWireFormatFactory
 - activemq::wireformat::openwire::OpenWireFormatFactory, 2712
- OpenWireFormatNegotiator
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2714
- OpenWireResponseBuilder
 - activemq::wireformat::openwire::OpenWireResponseBuilder, 2718
- operationType
 - activemq::commands::DestinationInfo, 1618
- operator<
 - activemq::commands::BrokerId, 800
 - activemq::commands::ConnectionId, 1233
 - activemq::commands::ConsumerId, 1333
 - activemq::commands::LocalTransactionId, 2203
 - activemq::commands::MessageId, 2498
 - activemq::commands::ProducerId, 2873
 - activemq::commands::SessionId, 3164
 - activemq::commands::TransactionId, 3572
 - activemq::commands::XATransactionId, 3768
 - decaf::lang::Boolean, 783
 - decaf::lang::Byte, 889
 - decaf::lang::Character, 1024, 1025
 - decaf::lang::Comparable, 1126
 - decaf::lang::Double, 1679
 - decaf::lang::Float, 1787
 - decaf::lang::Integer, 1949
 - decaf::lang::Long, 2275
 - decaf::lang::Short, 3225
 - decaf::net::URI, 3669
 - decaf::nio::ByteBuffer, 970
 - decaf::nio::CharBuffer, 1047
 - decaf::nio::DoubleBuffer, 1699
 - decaf::nio::FloatBuffer, 1807
 - decaf::nio::IntBuffer, 1937
 - decaf::nio::LongBuffer, 2298
 - decaf::nio::ShortBuffer, 3245
 - decaf::util::concurrent::TimeUnit, 3562
 - decaf::util::Date, 1561
 - decaf::util::logging::Level, 2188
 - decaf::util::UUID, 3710
- operator*
 - decaf::lang::Pointer, 2761
- operator()
 - decaf::lang::ArrayPointerComparator, 677
 - decaf::lang::ArrayPointerComparator, 2764
 - decaf::util::Comparator, 1129
 - decaf::util::comparators::Less, 2183
 - std::less< decaf::lang::ArrayPointer< T > >, 2184
 - std::less< decaf::lang::Pointer< T > >, 2185
 - operator->
 - decaf::lang::Pointer, 2761
 - activemq::cmsutil::CachedConsumer, 995
 - activemq::cmsutil::CachedProducer, 999
 - activemq::cmsutil::CmsAccessor, 1071
 - activemq::cmsutil::CmsDestinationAccessor, 1074
 - activemq::cmsutil::CmsTemplate, 1090
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2869
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2972
 - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3071
 - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3072
 - activemq::cmsutil::CmsTemplate::SendExecutor, 3136
 - activemq::cmsutil::DynamicDestinationResolver, 1705
 - activemq::cmsutil::PooledSession, 2776
 - activemq::cmsutil::ResourceLifecycleManager, 3076
 - activemq::cmsutil::SessionPool, 3217
 - activemq::commands::BrokerId, 800
 - activemq::commands::ConnectionId, 1233
 - activemq::commands::ConsumerId, 1333
 - activemq::commands::LocalTransactionId, 2203
 - activemq::commands::MessageId, 2498
 - activemq::commands::ProducerId, 2873
 - activemq::commands::SessionId, 3164
 - activemq::commands::TransactionId, 3572
 - activemq::commands::XATransactionId, 3768
 - activemq::library::ActiveMQCPP, 279
 - activemq::util::PrimitiveValueNode, 2827
 - decaf::lang::ArrayPointer, 674
 - decaf::lang::Exception, 1717
 - decaf::lang::Pointer, 2762
 - decaf::util::AbstractCollection, 151
 - decaf::util::Date, 1562
 - decaf::util::logging::LogManager, 2259
 - decaf::util::PriorityQueue, 2837
 - decaf::util::Properties, 2933

- operator==
 - activemq::commands::BrokerId, 800
 - activemq::commands::ConnectionId, 1233
 - activemq::commands::ConsumerId, 1333
 - activemq::commands::LocalTransactionId, 2203
 - activemq::commands::MessageId, 2498
 - activemq::commands::ProducerId, 2873
 - activemq::commands::SessionId, 3164
 - activemq::commands::TransactionId, 3572
 - activemq::commands::XATransactionId, 3768
 - activemq::util::PrimitiveValueNode, 2827
- decaf::lang, 129, 130
 - decaf::lang::ArrayPointer, 674, 676
 - decaf::lang::Boolean, 784
 - decaf::lang::Byte, 889, 890
 - decaf::lang::Character, 1025
 - decaf::lang::Comparable, 1127
 - decaf::lang::Double, 1679, 1680
 - decaf::lang::Float, 1787, 1788
 - decaf::lang::Integer, 1949
 - decaf::lang::Long, 2275
 - decaf::lang::Pointer, 2762, 2763
 - decaf::lang::Short, 3225, 3226
 - decaf::net::URI, 3669
 - decaf::nio::ByteBuffer, 970
 - decaf::nio::CharBuffer, 1047
 - decaf::nio::DoubleBuffer, 1700
 - decaf::nio::FloatBuffer, 1807
 - decaf::nio::IntBuffer, 1938
 - decaf::nio::LongBuffer, 2298
 - decaf::nio::ShortBuffer, 3246
 - decaf::util::concurrent::TimeUnit, 3563
 - decaf::util::Date, 1562
 - decaf::util::logging::Level, 2188
 - decaf::util::UUID, 3710
- operator[]
 - activemq::wireformat::openwire::utils::HexTable, 1858
 - decaf::internal::util::ByteArrayAdapter, 908
 - decaf::lang::ArrayPointer, 674, 675
- opt_len
 - internal_state, 1985
- optimizedAcknowledge
 - activemq::commands::ConsumerInfo, 1365
- options
 - activemq::commands::ActiveMQDestination, 289
- ordered
 - activemq::commands::ActiveMQDestination, 289
- orderedTarget
 - activemq::commands::ActiveMQDestination, 289
- originalDestination
 - activemq::commands::Message, 2374
- originalTransactionId
 - activemq::commands::Message, 2374
- OS
 - inflate.h, 4205
- os
 - gz_header_s, 1849
- OS_CODE
 - zutil.h, 4217
- out
 - decaf::io::FileDescriptor, 1769
 - gz_state, 1851
- OutputStream
 - decaf::io::OutputStream, 2720
- outputStream
 - decaf::io::FilterOutputStream, 1780
- OutputStreamWriter
 - decaf::io::OutputStreamWriter, 2727
- own
 - decaf::io::FilterInputStream, 1777
 - decaf::io::FilterOutputStream, 1780
- ownDeflater
 - decaf::util::zip::DeflaterOutputStream, 1608
- ownInflater
 - decaf::util::zip::InflaterInputStream, 1909
- PARAM_CLIENTID
 - activemq::core::ActiveMQConstants, 268
- PARAM_PASSWORD
 - activemq::core::ActiveMQConstants, 268
- PARAM_USERNAME
 - activemq::core::ActiveMQConstants, 268
- parent
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2869
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2972
- park
 - decaf::util::concurrent::locks::LockSupport, 2236
- parkNanos
 - decaf::util::concurrent::locks::LockSupport, 2236
- parkUntil
 - decaf::util::concurrent::locks::LockSupport, 2237
- parse
 - decaf::internal::util::HexStringParser, 1856
 - decaf::util::logging::Level, 2188
- parseAuthority

- decaf::internal::net::URIHelper, 3678
- parseBoolean
 - decaf::lang::Boolean, 784
- parseByte
 - decaf::lang::Byte, 890
- parseComposite
 - activemq::util::URISupport, 3685
- parseDouble
 - decaf::internal::util::HexStringParser, 1856
 - decaf::lang::Double, 1680
- parseFloat
 - decaf::internal::util::HexStringParser, 1857
 - decaf::lang::Float, 1788
- parseInt
 - decaf::lang::Integer, 1950
- parseLong
 - decaf::lang::Long, 2276
- parseQuery
 - activemq::util::URISupport, 3685
- parseServerAuthority
 - decaf::net::URI, 3669
- parseShort
 - decaf::lang::Short, 3226
- parseURI
 - decaf::internal::net::URIHelper, 3678
- parseURL
 - activemq::util::URISupport, 3686
- PartialCommand
 - activemq::commands::PartialCommand, 2729
- PartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2753
 - activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2736
 - activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2745
 - activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2749
 - activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2741
 - activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2732
- password
 - activemq::commands::ConnectionInfo, 1263
- path
 - gz_state, 1851
- peek
 - activemq::core::MessageDispatchChannel, 2435
 - decaf::internal::util::TimerTaskHeap, 3558
 - decaf::util::concurrent::SynchronousQueue, 3484
- decaf::util::PriorityQueue, 2837
- decaf::util::Queue, 2949
- peerBrokerInfos
 - activemq::commands::BrokerInfo, 831
- pending
 - internal_state, 1985
- pending_buf
 - internal_state, 1985
- pending_buf_size
 - internal_state, 1985
- pending_out
 - internal_state, 1985
- PERSISTENT
 - cms::DeliveryMode, 1610
- persistent
 - activemq::commands::Message, 2374
- physicalName
 - activemq::commands::ActiveMQDestination, 289
- PI
 - decaf::lang::Math, 2354
- Pointer
 - decaf::lang::Pointer, 2758, 2759
- PointerType
 - decaf::lang::ArrayPointer, 672
 - decaf::lang::Pointer, 2758
- poll
 - decaf::util::concurrent::BlockingQueue, 779
 - decaf::util::concurrent::SynchronousQueue, 3484
 - decaf::util::PriorityQueue, 2837
- PooledSession
 - activemq::commands::PooledSession, 2767
- PooledThread
 - activemq::commands::PooledThread, 2778
- pop
 - decaf::util::SynchronousQueue, 3387
- port
 - decaf::net::SVMarshal, 3313
- PortUnreachableException
 - decaf::net::PortUnreachableException, 2781, 2782
- Pos
 - deflate.h, 4202
- pos
 - gz_state, 1851
- Posf
 - deflate.h, 4202
- position
 - decaf::nio::Buffer, 860
- POSITIVE_INFINITY
 - decaf::lang::Double, 1683
 - decaf::lang::Float, 1791

- pow
 - decaf::lang::Math, 2348
- prefetch
 - activemq::commands::ConsumerControl, 1306
- PrefetchPolicy
 - activemq::core::PrefetchPolicy, 2785
- prefetchSize
 - activemq::commands::ConsumerInfo, 1365
- PRESET_DICT
 - zutil.h, 4217
- prev
 - internal_state, 1985
- prev_length
 - internal_state, 1985
- prev_match
 - internal_state, 1985
- previous
 - decaf::util::ListIterator, 2199
- previousIndex
 - decaf::util::ListIterator, 2199
- PrimitiveList
 - activemq::util::PrimitiveList, 2790
- PrimitiveMap
 - activemq::util::PrimitiveMap, 2800, 2801
- PrimitiveType
 - activemq::util::PrimitiveValueNode, 2821
- PrimitiveTypesMarshaller
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2810
- PrimitiveValueConverter
 - activemq::util::PrimitiveValueConverter, 2817
- PrimitiveValueNode
 - activemq::util::PrimitiveValueNode, 2821–2823
- printStackTrace
 - cms::CMSException, 1077
 - decaf::lang::Exception, 1717
 - decaf::lang::Throwable, 3540
- priority
 - activemq::commands::ConsumerInfo, 1365
 - activemq::commands::Message, 2374
- PriorityQueue
 - decaf::util::PriorityQueue, 2834, 2835
- PriorityQueueIterator
 - decaf::util::PriorityQueue, 2839
- processBeginTransaction
 - activemq::state::CommandVisitor, 1115
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1296
- processBrokerError
 - activemq::state::CommandVisitor, 1115
 - activemq::state::CommandVisitorAdapter, 1123
- processBrokerInfo
 - activemq::state::CommandVisitor, 1115
 - activemq::state::CommandVisitorAdapter, 1123
- processCommitTransactionOnePhase
 - activemq::state::CommandVisitor, 1115
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1296
- processCommitTransactionTwoPhase
 - activemq::state::CommandVisitor, 1115
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1296
- processConnectionControl
 - activemq::state::CommandVisitor, 1115
 - activemq::state::CommandVisitorAdapter, 1123
- processConnectionError
 - activemq::state::CommandVisitor, 1116
 - activemq::state::CommandVisitorAdapter, 1123
- processConnectionInfo
 - activemq::state::CommandVisitor, 1116
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1297
- processConsumerControl
 - activemq::state::CommandVisitor, 1116
 - activemq::state::CommandVisitorAdapter, 1123
- processConsumerInfo
 - activemq::state::CommandVisitor, 1116
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1297
- processControlCommand
 - activemq::state::CommandVisitor, 1116
 - activemq::state::CommandVisitorAdapter, 1123
- processDestinationInfo
 - activemq::state::CommandVisitor, 1116
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1297
- processEndTransaction

- activemq::state::CommandVisitor, 1116
- activemq::state::CommandVisitorAdapter, 1123
- activemq::state::ConnectionStateTracker, 1297
- processFlushCommand
 - activemq::state::CommandVisitor, 1116
 - activemq::state::CommandVisitorAdapter, 1123
- processForgetTransaction
 - activemq::state::CommandVisitor, 1117
 - activemq::state::CommandVisitorAdapter, 1123
- processKeepAliveInfo
 - activemq::state::CommandVisitor, 1117
 - activemq::state::CommandVisitorAdapter, 1123
- processMessage
 - activemq::state::CommandVisitor, 1117
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1297
- processMessageAck
 - activemq::state::CommandVisitor, 1117
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1297
- processMessageDispatch
 - activemq::state::CommandVisitor, 1117
 - activemq::state::CommandVisitorAdapter, 1123
- processMessageDispatchNotification
 - activemq::state::CommandVisitor, 1117
 - activemq::state::CommandVisitorAdapter, 1123
- processMessagePull
 - activemq::state::CommandVisitor, 1117
 - activemq::state::CommandVisitorAdapter, 1123
- processPrepareTransaction
 - activemq::state::CommandVisitor, 1117
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1297
- processProducerAck
 - activemq::state::CommandVisitor, 1117
 - activemq::state::CommandVisitorAdapter, 1123
- processProducerInfo
 - activemq::state::CommandVisitor, 1118
- activemq::state::CommandVisitorAdapter, 1123
- activemq::state::ConnectionStateTracker, 1298
- processRecoverTransactions
 - activemq::state::CommandVisitor, 1118
 - activemq::state::CommandVisitorAdapter, 1123
- processRemoveConnection
 - activemq::state::CommandVisitor, 1118
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1298
- processRemoveConsumer
 - activemq::state::CommandVisitor, 1118
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1298
- processRemoveDestination
 - activemq::state::CommandVisitor, 1118
 - activemq::state::CommandVisitorAdapter, 1123
 - activemq::state::ConnectionStateTracker, 1298
- processRemoveInfo
 - activemq::state::CommandVisitor, 1118
 - activemq::state::CommandVisitorAdapter, 1123
- processRemoveProducer
 - activemq::state::CommandVisitor, 1118
 - activemq::state::CommandVisitorAdapter, 1124
 - activemq::state::ConnectionStateTracker, 1298
- processRemoveSession
 - activemq::state::CommandVisitor, 1118
 - activemq::state::CommandVisitorAdapter, 1124
 - activemq::state::ConnectionStateTracker, 1298
- processRemoveSubscriptionInfo
 - activemq::state::CommandVisitor, 1119
 - activemq::state::CommandVisitorAdapter, 1124
- processReplayCommand
 - activemq::state::CommandVisitor, 1119
 - activemq::state::CommandVisitorAdapter, 1124
- processResponse
 - activemq::state::CommandVisitor, 1119
 - activemq::state::CommandVisitorAdapter, 1124

- processRollbackTransaction
 - activemq::state::CommandVisitor, 1119
 - activemq::state::CommandVisitorAdapter, 1124
 - activemq::state::ConnectionStateTracker, 1298
- processSessionInfo
 - activemq::state::CommandVisitor, 1119
 - activemq::state::CommandVisitorAdapter, 1124
 - activemq::state::ConnectionStateTracker, 1299
- processShutdownInfo
 - activemq::state::CommandVisitor, 1119
 - activemq::state::CommandVisitorAdapter, 1124
- processTransactionInfo
 - activemq::state::CommandVisitor, 1119
 - activemq::state::CommandVisitorAdapter, 1124
- processWireFormat
 - activemq::state::CommandVisitor, 1119
 - activemq::state::CommandVisitorAdapter, 1125
- PRODUCER_ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 289
- ProducerAck
 - activemq::commands::ProducerAck, 2840
- ProducerAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 2864
 - activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 2844
 - activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 2852
 - activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 2848
 - activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 2856
 - activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 2860
- ProducerExecutor
 - activemq::cmsutil::CmsTemplate, 1096
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2868
- ProducerId
 - activemq::commands::ProducerId, 2871
- producerId
 - activemq::commands::Message, 2374
 - activemq::commands::MessageId, 2499
 - activemq::commands::ProducerAck, 2842
 - activemq::commands::ProducerInfo, 2902
- ProducerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 2895
 - activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 2875
 - activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 2883
 - activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 2879
 - activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 2887
 - activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 2891
- ProducerInfo
 - activemq::commands::ProducerInfo, 2899
- ProducerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 2911
 - activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 2907
 - activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 2919
 - activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 2903
 - activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2915
 - activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 2923
- producerSequenceId
 - activemq::commands::MessageId, 2499
- ProducerState
 - activemq::state::ProducerState, 2926
- ProducerStateMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerStateMarshaller, 2929
- Properties
 - activemq::commands::ActiveMQMessageTemplate, 390
- propertyExists
 - activemq::state::ProducerState, 2926
- propertyNames
 - activemq::state::ProducerState, 2926
- ProtocolException
 - activemq::state::ProducerState, 2926
- ProtocolExceptionMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProtocolExceptionMarshaller, 2938
 - activemq::wireformat::openwire::marshal::v2::ProtocolExceptionMarshaller, 2939
- providerGenerateSeed
 - decaf::internal::security::SecureRandomImpl, 3122, 3123
 - decaf::security::SecureRandomSpi, 3125
- providerGetDefaultSSLParameters
 - decaf::net::ssl::SSLContextSpi, 3324
- providerGetServerSocketFactory
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2660
 - decaf::net::ssl::SSLContextSpi, 3325
- providerGetSocketFactory
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2660

- decaf::net::ssl::SSLContextSpi, 3325
- providerGetSupportedSSLParameters
 - decaf::net::ssl::SSLContextSpi, 3325
- providerInit
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2660
 - decaf::net::ssl::SSLContextSpi, 3326
- providerNextBytes
 - decaf::internal::security::SecureRandomImpl, 3123
 - decaf::security::SecureRandomSpi, 3125
- providerSetSeed
 - decaf::internal::security::SecureRandomImpl, 3123, 3124
 - decaf::security::SecureRandomSpi, 3126
- publish
 - decaf::util::logging::ConsoleHandler, 1301
 - decaf::util::logging::Handler, 1854
 - decaf::util::logging::StreamHandler, 3414
- purge
 - decaf::util::Timer, 3545
- push
 - decaf::util::StlQueue, 3387
- PushbackInputStream
 - decaf::io::PushbackInputStream, 2942
- put
 - decaf::internal::nio::ByteBuffer, 937
 - decaf::internal::nio::CharArrayBuffer, 1035
 - decaf::internal::nio::DoubleArrayBuffer, 1691
 - decaf::internal::nio::FloatArrayBuffer, 1799
 - decaf::internal::nio::IntArrayBuffer, 1929
 - decaf::internal::nio::LongArrayBuffer, 2289, 2290
 - decaf::internal::nio::ShortArrayBuffer, 3237
 - decaf::internal::util::ByteArrayAdapter, 909
 - decaf::nio::ByteBuffer, 970–972
 - decaf::nio::CharBuffer, 1047–1050
 - decaf::nio::DoubleBuffer, 1700, 1701
 - decaf::nio::FloatBuffer, 1807–1809
 - decaf::nio::IntBuffer, 1938, 1939
 - decaf::nio::LongBuffer, 2298–2300
 - decaf::nio::ShortBuffer, 3246, 3247
 - decaf::util::concurrent::BlockingQueue, 779
 - decaf::util::concurrent::ConcurrentStlMap, 1149
 - decaf::util::concurrent::SynchronousQueue, 3484
 - decaf::util::Map, 2313
 - decaf::util::StlMap, 3378
- put_byte
 - deflate.h, 4202
- putAll
 - decaf::util::concurrent::ConcurrentStlMap, 1150
 - decaf::util::Map, 2314
 - decaf::util::StlMap, 3378
- putChar
 - decaf::internal::nio::ByteBuffer, 938
 - decaf::internal::util::ByteArrayAdapter, 909
 - decaf::nio::ByteBuffer, 972, 973
- putDouble
 - decaf::internal::nio::ByteBuffer, 939
 - decaf::internal::util::ByteArrayAdapter, 909
 - decaf::nio::ByteBuffer, 973, 974
- putDoubleAt
 - decaf::internal::util::ByteArrayAdapter, 910
- putFloat
 - decaf::internal::nio::ByteBuffer, 939, 940
 - decaf::internal::util::ByteArrayAdapter, 910
 - decaf::nio::ByteBuffer, 974
- putFloatAt
 - decaf::internal::util::ByteArrayAdapter, 911
- putIfAbsent
 - decaf::util::concurrent::ConcurrentMap, 1137
 - decaf::util::concurrent::ConcurrentStlMap, 1150
- putInt
 - decaf::internal::nio::ByteBuffer, 940, 941
 - decaf::internal::util::ByteArrayAdapter, 911
 - decaf::nio::ByteBuffer, 975
- putIntAt
 - decaf::internal::util::ByteArrayAdapter, 911
- putLong
 - decaf::internal::nio::ByteBuffer, 941, 942
 - decaf::internal::util::ByteArrayAdapter, 912
 - decaf::nio::ByteBuffer, 976
- putLongAt
 - decaf::internal::util::ByteArrayAdapter, 912
- putShort
 - decaf::internal::nio::ByteBuffer, 942
 - decaf::internal::util::ByteArrayAdapter, 913

- decaf::nio::ByteBuffer, 976, 977
- putShortAt
 - decaf::internal::util::ByteArrayAdapter, 913
- QUEUE
 - cms::Destination, 1611
- QUEUE_PREFIX
 - activemq::wireformat::stomp::StompCommand, 3397
- QUEUE_QUALIFIED_PREFIX
 - activemq::commands::ActiveMQDestination, 289
- queueTask
 - decaf::util::concurrent::ThreadPool, 3535
- quoteIllegal
 - decaf::internal::net::URLEncoderDecoder, 3673
- Random
 - decaf::util::Random, 2954
- random
 - decaf::lang::Math, 2348
- randomUUID
 - decaf::util::UUID, 3711
- raw
 - gz_state, 1851
- reached
 - decaf::net::InetAddress, 1891
- read
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2682
 - decaf::internal::net::tcp::TcpSocket, 3504
 - decaf::internal::util::ByteArrayAdapter, 913
 - decaf::io::InputStream, 1914, 1915
 - decaf::io::Reader, 2963–2965
 - decaf::lang::Readable, 2958
 - decaf::nio::CharBuffer, 1050
- readAsciiString
 - activemq::wireformat::openwire::marshal::BaseDataStreamHandler, 753
- readBoolean
 - activemq::commands::ActiveMQBytesMessage, 200
 - activemq::commands::ActiveMQStreamMessage, 490
 - activemq::wireformat::openwire::utils::BooleanStream, 789
 - cms::BytesMessage, 983
 - cms::StreamMessage, 3418
 - decaf::io::DataInput, 1454
 - decaf::io::DataInputStream, 1462
- readByte
 - activemq::commands::ActiveMQBytesMessage, 200
 - activemq::commands::ActiveMQStreamMessage, 490
 - cms::BytesMessage, 983
 - cms::StreamMessage, 3418
 - decaf::io::DataInput, 1454
 - decaf::io::DataInputStream, 1462
- readChar
 - activemq::commands::ActiveMQBytesMessage, 202
 - activemq::commands::ActiveMQStreamMessage, 492
 - cms::BytesMessage, 985
 - cms::StreamMessage, 3420
 - decaf::io::DataInput, 1454
 - decaf::io::DataInputStream, 1463
- ReadChecker
 - activemq::transport::inactivity::InactivityMonitor, 1877
 - activemq::transport::inactivity::ReadChecker, 2960
- readConfiguration
 - decaf::util::logging::LogManager, 2259, 2260
- readDouble
 - activemq::commands::ActiveMQBytesMessage, 202
 - activemq::commands::ActiveMQStreamMessage, 492
 - cms::BytesMessage, 985
 - cms::StreamMessage, 3420
 - decaf::io::DataInput, 1455
 - decaf::io::DataInputStream, 1463
- Reader
 - decaf::io::Reader, 2962
- readFloat
 - activemq::commands::ActiveMQBytesMessage, 202
 - activemq::commands::ActiveMQStreamMessage, 493
 - cms::BytesMessage, 986
 - cms::StreamMessage, 3421
 - decaf::io::DataInput, 1455
 - decaf::io::DataInputStream, 1463
- readFully
 - decaf::io::DataInput, 1455, 1456

- decaf::io::DataInputStream, 1464
- readInt
 - activemq::commands::ActiveMQBytesMessage, 203
 - activemq::commands::ActiveMQStreamMessage, 493
 - cms::BytesMessage, 986
 - cms::StreamMessage, 3421
 - decaf::io::DataInput, 1457
 - decaf::io::DataInputStream, 1465
- readLine
 - decaf::io::DataInput, 1457
 - decaf::io::DataInputStream, 1465
- readLock
 - decaf::util::concurrent::locks::ReadWriteLock, 2970
- readLong
 - activemq::commands::ActiveMQBytesMessage, 203
 - activemq::commands::ActiveMQStreamMessage, 493
 - cms::BytesMessage, 986
 - cms::StreamMessage, 3422
 - decaf::io::DataInput, 1457
 - decaf::io::DataInputStream, 1466
- readOnly
 - decaf::internal::nio::CharArrayBuffer, 1037
- readonly
 - decaf::io::FileDescriptor, 1770
- ReadOnlyBufferException
 - decaf::nio::ReadOnlyBufferException, 2967, 2968
- readShort
 - activemq::commands::ActiveMQBytesMessage, 203
 - activemq::commands::ActiveMQStreamMessage, 494
 - cms::BytesMessage, 987
 - cms::StreamMessage, 3422
 - decaf::io::DataInput, 1458
 - decaf::io::DataInputStream, 1466
- readString
 - activemq::commands::ActiveMQBytesMessage, 204
 - activemq::commands::ActiveMQStreamMessage, 494
 - cms::BytesMessage, 987
 - cms::StreamMessage, 3422
 - decaf::io::DataInput, 1458
 - decaf::io::DataInputStream, 1466
- readString16
 - activemq::util::MarshallingSupport, 2337
- readString32
 - activemq::util::MarshallingSupport, 2337
- readUnsignedByte
 - decaf::io::DataInput, 1458
 - decaf::io::DataInputStream, 1467
- readUnsignedShort
 - activemq::commands::ActiveMQBytesMessage, 204
 - activemq::commands::ActiveMQStreamMessage, 495
 - cms::BytesMessage, 987
 - cms::StreamMessage, 3423
 - decaf::io::DataInput, 1459
 - decaf::io::DataInputStream, 1467
- readUTF
 - activemq::commands::ActiveMQBytesMessage, 204
 - cms::BytesMessage, 988
 - decaf::io::DataInput, 1459
 - decaf::io::DataInputStream, 1467
- ready
 - decaf::io::InputStreamReader, 1921
 - decaf::io::Reader, 2965
- rebalanceConnection
 - activemq::commands::ConnectionControl, 1176
- RECEIPT
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- receive
 - activemq::cmsutil::CachedConsumer, 995
 - activemq::cmsutil::CmsTemplate, 1090, 1091
 - activemq::core::ActiveMQConsumer, 275
 - cms::MessageConsumer, 2424, 2425
- RECEIVE_TIMEOUT_INDEFINITE_WAIT
 - activemq::cmsutil::CmsTemplate, 1096
- RECEIVE_TIMEOUT_NO_WAIT
 - activemq::cmsutil::CmsTemplate, 1096
- ReceiveExecutor
 - activemq::cmsutil::CmsTemplate, 1096
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2971
- receiveNoWait
 - activemq::cmsutil::CachedConsumer, 995
 - activemq::core::ActiveMQConsumer, 276
 - cms::MessageConsumer, 2425
- receiveSelected
 - activemq::cmsutil::CmsTemplate, 1091, 1092
- recievedByDFBBridge
 - activemq::commands::Message, 2374
- reconnect
 - activemq::transport::failover::FailoverTransport, 1759
 - activemq::transport::IOTransport, 2009

- activemq::transport::mock::MockTransport, 2598
- activemq::transport::Transport, 3632
- activemq::transport::TransportFilter, 3641
- reconnectTo
 - activemq::commands::ConnectionControl, 1176
- recover
 - activemq::cmsutil::PooledSession, 2776
 - activemq::core::ActiveMQSession, 479
 - cms::Session, 3159
- redeliveryCounter
 - activemq::commands::Message, 2374
 - activemq::commands::MessageDispatch, 2431
- RedeliveryPolicy
 - activemq::core::RedeliveryPolicy, 2974
- redispatch
 - activemq::core::ActiveMQSession, 479
- ReentrantLock
 - decaf::util::concurrent::locks::ReentrantLock, 2979
- ReferenceType
 - decaf::lang::ArrayPointer, 672
 - decaf::lang::Pointer, 2758
- registerFactory
 - activemq::transport::TransportRegistry, 3647
 - activemq::wireformat::WireFormatRegistry, 3754
- rejectedExecution
 - decaf::util::concurrent::RejectedExecutionHandler, 2987
- RejectedExecutionException
 - decaf::util::concurrent::RejectedExecutionException, 2985, 2986
- relativize
 - decaf::net::URI, 3670
- release
 - decaf::lang::ArrayPointer, 675
 - decaf::lang::Pointer, 2762
 - decaf::util::concurrent::atomic::AtomicRefCount, 686
 - decaf::util::concurrent::Semaphore, 3132
- releaseAll
 - activemq::cmsutil::ResourceLifecycleManager, 3076
- remaining
 - decaf::nio::Buffer, 860
- remainingCapacity
 - decaf::util::concurrent::BlockingQueue, 780
 - decaf::util::concurrent::SynchronousQueue, 3485
- remove
 - activemq::util::ActiveMQProperties, 434
 - cms::CMSProperties, 1081
 - decaf::internal::util::TimerTaskHeap, 3558
 - decaf::util::AbstractCollection, 151
 - decaf::util::AbstractQueue, 161
 - decaf::util::Collection, 1104
 - decaf::util::concurrent::ConcurrentMap, 1138
 - decaf::util::concurrent::ConcurrentStlMap, 1151, 1152
 - decaf::util::concurrent::SynchronousQueue, 3485
 - decaf::util::Iterator, 2013
 - decaf::util::List, 2196
 - decaf::util::Map, 2315
 - decaf::util::PriorityQueue, 2838
 - decaf::util::Properties, 2933
 - decaf::util::Queue, 2950
 - decaf::util::StlList, 3367, 3368
 - decaf::util::StlMap, 3379
 - decaf::util::StlSet, 3394
- removeAll
 - activemq::core::MessageDispatchChannel, 2435
 - decaf::util::AbstractCollection, 152
 - decaf::util::AbstractSet, 163
 - decaf::util::Collection, 1105
 - decaf::util::concurrent::SynchronousQueue, 3486
- removeConsumer
 - activemq::core::ActiveMQSession, 479
- activemq::state::SessionState, 3219
- removeDispatcher
 - activemq::core::ActiveMQConnection, 246
- removeHandler
 - decaf::util::logging::Logger, 2247
- RemoveInfo
 - activemq::commands::RemoveInfo, 2989
- RemoveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 3004
 - activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 2992
 - activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 3000
 - activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 3012
 - activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 3008
 - activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 2996
- removeProducer
 - activemq::core::ActiveMQConnection, 246
 - activemq::core::ActiveMQSession, 480

- activemq::state::SessionState, 3219
- removeProperty
 - activemq::wireformat::stomp::StompFrame, 3401
- removePropertyChangeListener
 - decaf::util::logging::LogManager, 2260
- removeSession
 - activemq::core::ActiveMQConnection, 246
 - activemq::state::ConnectionState, 1293
- RemoveSubscriptionInfo
 - activemq::commands::RemoveSubscriptionInfo, 3016
- RemoveSubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3020
 - activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3028
 - activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3024
 - activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3040
 - activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3036
 - activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3032
- removeSynchronization
 - activemq::core::ActiveMQTransactionContext, 663
- removeTask
 - activemq::threads::CompositeTaskRunner, 1134
- removeTempDestination
 - activemq::state::ConnectionState, 1293
- RemoveTransactionAction
 - activemq::state::ConnectionStateTracker, 1300
- removeTransactionState
 - activemq::state::ConnectionState, 1293
- removeTransportListener
 - activemq::core::ActiveMQConnection, 247
- removeURI
 - activemq::transport::CompositeTransport, 1136
 - activemq::transport::failover::FailoverTransport, 1759
 - activemq::transport::failover::URIPool, 3683
- renegotiateWireFormat
 - activemq::wireformat::openwire::OpenWireFormat, 2708
- replace
 - decaf::util::concurrent::ConcurrentMap, 1139
 - decaf::util::concurrent::ConcurrentStlMap, 1152, 1153
- ReplayCommand
 - activemq::commands::ReplayCommand, 3044
- ReplayCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 3051
 - activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 3055
 - activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 3059
 - activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 3047
 - activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 3067
 - activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3063
- replyTo
 - activemq::commands::Message, 2374
- reportError
 - decaf::util::logging::Handler, 1854
- request
 - activemq::transport::correlator::ResponseCorrelator, 3083, 3084
- activemq::transport::failover::FailoverTransport, 1760
- activemq::transport::IOTransport, 2009, 2010
- activemq::transport::logging::LoggingTransport, 2254
- activemq::transport::mock::MockTransport, 2598, 2599
- activemq::transport::Transport, 3632, 3633
- activemq::transport::TransportFilter, 3641, 3642
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2715, 2716
- reserve
 - decaf::util::concurrent::ThreadPool, 3535
- reserved
 - z_stream_s, 3795
- reset
 - activemq::commands::ActiveMQBytesMessage, 205
 - activemq::commands::ActiveMQStreamMessage, 495
 - activemq::state::ConnectionState, 1293
 - cms::BytesMessage, 988
 - decaf::internal::util::TimerTaskHeap, 3558
 - decaf::io::BufferedInputStream, 865
 - decaf::io::ByteArrayInputStream, 949
 - decaf::io::ByteArrayOutputStream, 952
 - decaf::io::FilterInputStream, 1775

- decaf::io::InputStream, 1916
- decaf::io::PushbackInputStream, 2944
- decaf::io::Reader, 2965
- decaf::lang::ArrayPointer, 675
- decaf::lang::Pointer, 2762
- decaf::nio::Buffer, 860
- decaf::util::logging::LogManager, 2260
- decaf::util::StringTokenizer, 3432
- decaf::util::zip::Adler32, 664
- decaf::util::zip::Checksum, 1060
- decaf::util::zip::CRC32, 1421
- decaf::util::zip::Deflater, 1600
- decaf::util::zip::Inflater, 1899
- decaf::util::zip::InflaterInputStream, 1907
- resize
 - decaf::internal::util::ByteArrayAdapter, 914
- resolve
 - decaf::net::URI, 3670
- resolveDestinationName
 - activemq::cmsutil::CmsDestinationAccessor, 1074
 - activemq::cmsutil::DestinationResolver, 1643
 - activemq::cmsutil::DynamicDestinationResolver, 1705
- ResolveProducerExecutor
 - activemq::cmsutil::CmsTemplate, 1096
 - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3071
- ResolveReceiveExecutor
 - activemq::cmsutil::CmsTemplate, 1096
 - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3072
- ResourceLifecycleManager
 - activemq::cmsutil::ResourceLifecycleManager, 3075
 - decaf::internal::util::ResourceLifecycleManager, 3073
- Response
 - activemq::commands::Response, 3077
- ResponseCorrelator
 - activemq::transport::correlator::ResponseCorrelator, 3082
- ResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 3104
 - activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 3090
 - activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 3099
 - activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 3086
- activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 3095
- activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3108
- restore
 - activemq::state::ConnectionStateTracker, 1299
- restoreTransport
 - activemq::transport::failover::FailoverTransport, 1760
- result
 - activemq::commands::IntegerResponse, 1958
- resume
 - activemq::commands::ConnectionControl, 1176
- retainAll
 - decaf::util::AbstractCollection, 153
 - decaf::util::Collection, 1105
 - decaf::util::concurrent::SynchronousQueue, 3486
- retroactive
 - activemq::commands::ConsumerInfo, 1365
- returnInstance
- returnSession
 - decaf::util::logging::LogWriter, 2267
 - activemq::cmsutil::SessionPool, 3217
- reverse
 - decaf::lang::Integer, 1951
 - decaf::lang::Long, 2277
 - decaf::util::StlQueue, 3387
- reverseBytes
 - decaf::lang::Integer, 1951
 - decaf::lang::Long, 2277
 - decaf::lang::Short, 3227
- rewind
 - decaf::nio::Buffer, 861
- rollback
 - activemq::cmsutil::PooledSession, 2776
 - activemq::core::ActiveMQConsumer, 276
 - activemq::core::ActiveMQSession, 480
 - activemq::core::ActiveMQTransactionContext, 663
 - cms::Session, 3159
- rotateLeft
 - decaf::lang::Integer, 1951
 - decaf::lang::Long, 2277
- rotateRight
 - decaf::lang::Integer, 1952
 - decaf::lang::Long, 2277
- round
 - decaf::lang::Math, 2349
- run

- activemq::threads::CompositeTaskRunner, 1134
- activemq::threads::DedicatedTaskRunner, 1565
- activemq::transport::inactivity::ReadChecker, 2960
- activemq::transport::inactivity::WriteChecker, 3755
- activemq::transport::IOTransport, 2010
- activemq::transport::mock::InternalCommandListener, 1093
- decaf::lang::Runnable, 3112
- decaf::lang::Thread, 3527
- decaf::util::concurrent::PooledThread, 2778
- RUNNABLE
 - decaf::lang::Thread, 3523
- RuntimeException
 - decaf::lang::exceptions::RuntimeException, 3115
- sane
 - inflate_state, 1893
- schedule
 - decaf::util::Timer, 3546–3550
- scheduleAtFixedRate
 - decaf::util::Timer, 3551–3553
- scheduledExecutionTime
 - decaf::util::TimerTask, 3555
- SECONDS
 - decaf::util::concurrent::TimeUnit, 3567
- SecureRandom
 - decaf::security::SecureRandom, 3118
- SecureRandomImpl
 - decaf::internal::security::SecureRandomImpl, 3122
- SecureRandomSpi
 - decaf::security::SecureRandomSpi, 3125
- seek
 - gz_state, 1851
- SEEK_CUR
 - zconf.h, 4211
- SEEK_END
 - zconf.h, 4211
- SEEK_SET
 - zconf.h, 4211
- selector
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2972
 - activemq::commands::ConsumerInfo, 1365
 - activemq::commands::SubscriptionInfo, 3438
- Semaphore
 - decaf::util::concurrent::Semaphore, 3129
- semaphore
 - decaf::util::concurrent::ConditionHandle, 1163
- SEND
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- send
 - activemq::cmsutil::CachedProducer, 1000, 1001
 - activemq::cmsutil::CmsTemplate, 1092, 1093
 - activemq::core::ActiveMQProducer, 427–429
 - activemq::core::ActiveMQSession, 480
 - cms::MessageProducer, 2553–2555
- SendExecutor
 - activemq::cmsutil::CmsTemplate, 1096
 - activemq::cmsutil::CmsTemplate::SendExecutor, 3136
- sendPullRequest
 - activemq::core::ActiveMQConnection, 247
- sendUrgentData
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2682
 - decaf::net::Socket, 3293
 - decaf::net::SocketImpl, 3312
- ServerSocket
 - decaf::net::ServerSocket, 3139, 3140
 - decaf::net::Socket, 3297
- ServerSocketFactory
 - decaf::net::ServerSocketFactory, 3146
- serviceName
 - activemq::commands::DiscoveryEvent, 1647
- SESSION_TRANSACTED
 - cms::Session, 3152
- SessionId
 - activemq::commands::SessionId, 3163
- sessionId
 - activemq::commands::ConsumerId, 1334
 - activemq::commands::ProducerId, 2873
 - activemq::commands::SessionInfo, 3191
- SessionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 3186
 - activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 3166
 - activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 3182
 - activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 3170
 - activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 3178
 - activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3174

- SessionInfo
 - activemq::commands::SessionInfo, 3189
- SessionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 3201
 - activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 3209
 - activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 3205
 - activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 3213
 - activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 3197
 - activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3193
- SessionPool
 - activemq::cmsutil::SessionPool, 3216
- SessionState
 - activemq::state::SessionState, 3219
- set
 - decaf::util::concurrent::atomic::AtomicBoolean, 679
 - decaf::util::concurrent::atomic::AtomicInteger, 684
 - decaf::util::concurrent::atomic::AtomicReference, 689
 - decaf::util::List, 2196
 - decaf::util::ListIterator, 2199
 - decaf::util::StlList, 3368
- setAbsolute
 - decaf::internal::net::URIType, 3695
- setAckHandler
 - activemq::commands::Message, 2369
- setAckMode
 - activemq::commands::SessionInfo, 3191
- setAckType
 - activemq::commands::MessageAck, 2397
- setAdditionalPredicate
 - activemq::commands::ConsumerInfo, 1363
- setAdvisory
 - activemq::commands::ActiveMQDestination, 287
- setAlwaysSyncSend
 - activemq::core::ActiveMQConnection, 247
 - activemq::core::ActiveMQConnectionFactory, 258
- setArrival
 - activemq::commands::Message, 2370
- setAuthority
 - decaf::internal::net::URIType, 3695
- setBackOffMultiplier
 - activemq::core::policies::DefaultRedeliveryPolicy, 1572
 - activemq::core::RedeliveryPolicy, 2976
- activemq::transport::failover::FailoverTransport, 1761
- setBackup
 - activemq::transport::failover::FailoverTransport, 1761
- setBackupPoolSize
 - activemq::transport::failover::BackupTransportPool, 691
- setBlocksize
 - activemq::transport::failover::FailoverTransport, 781
- setBlockSize
 - activemq::transport::failover::FailoverTransport, 781
- setBody
 - activemq::wireformat::stomp::StompFrame, 3401
- setBodyBytes
 - activemq::commands::ActiveMQBytesMessage, 205
- setBool
 - activemq::util::PrimitiveList, 2794
 - activemq::util::PrimitiveMap, 2805
 - activemq::util::PrimitiveValueNode, 2828
- setBoolean
 - activemq::commands::ActiveMQMapMessage, 325
- setBooleanProperty
 - cms::MapMessage, 2324
- setBooleanProperty
 - activemq::commands::ActiveMQMessageTemplate, 390
- setBranchQualifier
 - activemq::commands::XATransactionId, 3768
- setBrokerId
 - activemq::commands::BrokerInfo, 829
- setBrokerInTime
 - activemq::commands::Message, 2370
- setBrokerMasterConnector
 - activemq::commands::ConnectionInfo, 1262
- setBrokerName
 - activemq::commands::BrokerInfo, 829
- setBrokerOutTime
 - activemq::commands::Message, 2370
- setBrokerPath
 - activemq::commands::ConnectionInfo, 1262
- activemq::commands::ConsumerInfo, 1363

- activemq::commands::DestinationInfo, 1617
- activemq::commands::Message, 2370
- activemq::commands::ProducerInfo, 2900
- setBrokerSequenceId
 - activemq::commands::MessageId, 2498
- setBrokerUploadUrl
 - activemq::commands::BrokerInfo, 829
- setBrokerURL
 - activemq::commands::BrokerInfo, 829
 - activemq::core::ActiveMQConnection, 247
 - activemq::core::ActiveMQConnectionFactory, 259
- setBrowser
 - activemq::commands::ConsumerInfo, 1363
- setByte
 - activemq::commands::ActiveMQMapMessage, 325
 - activemq::util::PrimitiveList, 2795
 - activemq::util::PrimitiveMap, 2805
 - activemq::util::PrimitiveValueNode, 2828
 - cms::MapMessage, 2324
- setByteArray
 - activemq::util::PrimitiveList, 2795
 - activemq::util::PrimitiveMap, 2805
 - activemq::util::PrimitiveValueNode, 2828
 - decaf::io::BlockingByteArrayInputStream, 773
 - decaf::io::ByteArrayInputStream, 949, 950
- setByteProperty
 - activemq::commands::ActiveMQMessageTemplate, 391
 - activemq::wireformat::openwire::utils::MessageProperty, 2561
 - cms::Message, 2387
- setBytes
 - activemq::commands::ActiveMQMapMessage, 325
 - cms::MapMessage, 2324
- setCacheEnabled
 - activemq::commands::WireFormatInfo, 3724
 - activemq::wireformat::openwire::OpenWireFormat, 2708
- setCacheSize
 - activemq::commands::WireFormatInfo, 3725
 - activemq::wireformat::openwire::OpenWireFormat, 2708
- setCause
 - activemq::commands::BrokerError, 795
- setChar
 - activemq::commands::ActiveMQMapMessage, 326
- activemq::util::PrimitiveList, 2795
- activemq::util::PrimitiveMap, 2806
- activemq::util::PrimitiveValueNode, 2828
- cms::MapMessage, 2325
- setCipherSuites
 - decaf::net::ssl::SSLParameters, 3328
- setClientID
 - activemq::core::ActiveMQConnection, 247
 - cms::Connection, 1171
- setClientId
 - activemq::commands::ConnectionInfo, 1262
 - activemq::commands::JournalTopicAck, 2044
 - activemq::commands::RemoveSubscriptionInfo, 3018
 - activemq::commands::SubscriptionInfo, 3437
 - activemq::core::ActiveMQConnectionFactory, 259
- setClientMaster
 - activemq::commands::ConnectionInfo, 1262
- setClose
 - activemq::commands::ConnectionControl, 1175
 - activemq::commands::ConsumerControl, 1305
- setClosed
 - activemq::transport::failover::BackupTransport, 691
- setCloseTimeout
 - activemq::core::ActiveMQConnection, 248
 - activemq::core::ActiveMQConnectionFactory, 259
- setCluster
 - activemq::commands::Message, 2370
- setCMSCorrelationID
 - activemq::commands::ActiveMQMessageTemplate, 391
 - cms::Message, 2388
- setCMSDeliveryMode
 - activemq::commands::ActiveMQMessageTemplate, 391
 - cms::Message, 2388
- setCMSDestination
 - activemq::commands::ActiveMQMessageTemplate, 392
 - cms::Message, 2389
- setCMSExpiration
 - activemq::commands::ActiveMQMessageTemplate, 392
 - cms::Message, 2389
- setCMSMessageID
 - cms::Message, 2389

- activemq::commands::ActiveMQMessageTemplate, 392
- cms::Message, 2389
- setCMSPriority
 - activemq::commands::ActiveMQMessageTemplate, 392
 - cms::Message, 2390
- setCMSRedelivered
 - activemq::commands::ActiveMQMessageTemplate, 393
 - cms::Message, 2390
- setCMSReplyTo
 - activemq::commands::ActiveMQMessageTemplate, 393
 - cms::Message, 2390
- setCMSTimestamp
 - activemq::commands::ActiveMQMessageTemplate, 393
 - cms::Message, 2391
- setCMSType
 - activemq::commands::ActiveMQMessageTemplate, 394
 - cms::Message, 2391
- setCollisionAvoidancePercent
 - activemq::core::policies::DefaultRedeliveryPolicy, 1572
 - activemq::core::RedeliveryPolicy, 2976
- setCommand
 - activemq::commands::ControlCommand, 1392
 - activemq::wireformat::stomp::StompFrame, 3402
- setCommandId
 - activemq::commands::BaseCommand, 700
 - activemq::commands::Command, 1111
 - activemq::commands::PartialCommand, 2730
- setComponents
 - activemq::util::CompositeData, 1131
- setCompressed
 - activemq::commands::Message, 2370
- setConnectedBrokers
 - activemq::commands::ConnectionControl, 1175
- setConnection
 - activemq::commands::ActiveMQTempDestination, 526
 - activemq::commands::Message, 2370
- setConnectionFactory
 - activemq::cmsutil::CmsAccessor, 1071
- setConnectionId
 - activemq::commands::BrokerInfo, 829
 - activemq::commands::ConnectionError, 1203
- activemq::commands::ConnectionInfo, 1262
- activemq::commands::ConsumerId, 1333
- activemq::commands::DestinationInfo, 1617
- activemq::commands::LocalTransactionId, 2203
- activemq::commands::ProducerId, 2873
- activemq::commands::RemoveSubscriptionInfo, 3018
- activemq::commands::SessionId, 3164
- activemq::commands::TransactionInfo, 3597
- setConnectionInterruptProcessingComplete
 - activemq::state::ConnectionState, 1293
- activemq::transport::failover::FailoverTransport, 1761
- setConsumerId
 - activemq::commands::ConsumerControl, 1305
 - activemq::commands::ConsumerInfo, 1363
 - activemq::commands::MessageAck, 2398
 - activemq::commands::MessageDispatch, 2429
 - activemq::commands::MessageDispatchNotification, 2465
 - activemq::commands::MessagePull, 2567
- setContent
 - activemq::commands::Message, 2370
- setCorrelationId
 - activemq::commands::Message, 2371
 - activemq::commands::MessagePull, 2567
 - activemq::commands::Response, 3079
- setData
 - activemq::commands::DataArrayResponse, 1425
 - activemq::commands::DataResponse, 1479
 - activemq::commands::PartialCommand, 2730
- setDataStructure
 - activemq::commands::Message, 2371
- setDefault
 - decaf::net::ssl::SSLContext, 3323
- setDefaultClientId
 - activemq::core::ActiveMQConnection, 248
- setDefaultDestination
 - activemq::cmsutil::CmsTemplate, 1093
- setDefaultDestinationName
 - activemq::cmsutil::CmsTemplate, 1093
- setDeletedByBroker
 - activemq::commands::ActiveMQBlobMessage, 169
- setDeliveryMode
 - activemq::cmsutil::CachedProducer, 1001

- activemq::cmsutil::CmsTemplate, 1093
- activemq::core::ActiveMQProducer, 429
- cms::MessageProducer, 2555
- setDeliveryPersistent
 - activemq::cmsutil::CmsTemplate, 1094
- setDeliverySequenceId
 - activemq::commands::MessageDispatchNotification, 2465
- setDestination
 - activemq::commands::ConsumerControl, 1305
 - activemq::commands::ConsumerInfo, 1363
 - activemq::commands::DestinationInfo, 1617
 - activemq::commands::JournalQueueAck, 2016
 - activemq::commands::JournalTopicAck, 2044
 - activemq::commands::Message, 2371
 - activemq::commands::MessageAck, 2398
 - activemq::commands::MessageDispatch, 2430
 - activemq::commands::MessageDispatchNotification, 2465
 - activemq::commands::MessagePull, 2567
 - activemq::commands::ProducerInfo, 2901
 - activemq::commands::SubscriptionInfo, 3437
- setDestinationResolver
 - activemq::cmsutil::CmsDestinationAccessor, 1074
- setDictionary
 - decaf::util::zip::Deflater, 1600, 1601
 - decaf::util::zip::Inflater, 1899, 1900
- setDisableMessageID
 - activemq::cmsutil::CachedProducer, 1002
 - activemq::core::ActiveMQProducer, 429
 - cms::MessageProducer, 2556
- setDisableMessageTimeStamp
 - activemq::cmsutil::CachedProducer, 1002
 - activemq::core::ActiveMQProducer, 430
 - cms::MessageProducer, 2556
- setDispatchAsync
 - activemq::commands::ConsumerInfo, 1363
 - activemq::commands::ProducerInfo, 2901
 - activemq::core::ActiveMQConnection, 248
 - activemq::core::ActiveMQConnectionFactory, 259
- setDouble
 - activemq::commands::ActiveMQMapMessage, 326
 - activemq::util::PrimitiveList, 2796
 - activemq::util::PrimitiveMap, 2806
 - activemq::util::PrimitiveValueNode, 2828
- cms::MapMessage, 2325
- setDoubleProperty
 - activemq::commands::ActiveMQMessageTemplate, 394
 - activemq::wireformat::openwire::utils::MessagePropertyInterco, 2562
- cms::Message, 2392
- setDroppable
 - activemq::commands::Message, 2371
- setDuplexConnection
 - activemq::commands::BrokerInfo, 829
- setDurableTopicPrefetch
 - activemq::core::policies::DefaultPrefetchPolicy, 1568
 - activemq::core::PrefetchPolicy, 2786
- setEnabled
 - activemq::transport::failover::BackupTransportPool, 694
- setEnabledCipherSuites
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2663
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2668
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2683
 - decaf::net::ssl::SSLServerSocket, 3333
 - decaf::net::ssl::SSLSocket, 3342
- setEnabledProtocols
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2663
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2668
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2683
 - decaf::net::ssl::SSLServerSocket, 3334
 - decaf::net::ssl::SSLSocket, 3342
- setenv
 - decaf::lang::System, 3493
- setErrorHandler
 - decaf::util::logging::Handler, 1855
- setException
 - activemq::commands::ConnectionError, 1203
 - activemq::commands::ExceptionResponse, 1722
- setExceptionClass
 - activemq::commands::BrokerError, 796
- setExceptionListener
 - activemq::core::ActiveMQConnection, 248
 - activemq::core::ActiveMQConnectionFactory, 259
 - cms::Connection, 1171
- setExclusive

- activemq::commands::ActiveMQDestination, 287
- activemq::commands::ConsumerInfo, 1363
- setExit
 - activemq::commands::ConnectionControl, 1175
- setExpiration
 - activemq::commands::Message, 2371
- setExplicitQosEnabled
 - activemq::cmsutil::CmsTemplate, 1094
- setFailOnClose
 - activemq::transport::mock::MockTransport, 2599
- setFailOnKeepAliveSends
 - activemq::transport::mock::MockTransport, 2600
- setFailOnReceiveMessage
 - activemq::transport::mock::MockTransport, 2600
- setFailOnSendMessage
 - activemq::transport::mock::MockTransport, 2600
- setFailOnStart
 - activemq::transport::mock::MockTransport, 2600
- setFailOnStop
 - activemq::transport::mock::MockTransport, 2600
- setFault Tolerant
 - activemq::commands::ConnectionControl, 1175
 - activemq::commands::ConnectionInfo, 1262
- setFault Tolerant Configuration
 - activemq::commands::BrokerInfo, 829
- setFilter
 - decaf::util::logging::Handler, 1855
 - decaf::util::logging::Logger, 2247
- setFirstMessageId
 - activemq::commands::MessageAck, 2398
- setFirstNakNumber
 - activemq::commands::ReplayCommand, 3045
- setFloat
 - activemq::commands::ActiveMQMapMessage, 326
 - activemq::util::PrimitiveList, 2796
 - activemq::util::PrimitiveMap, 2806
 - activemq::util::PrimitiveValueNode, 2829
 - cms::MapMessage, 2325
- setFloatProperty
 - activemq::commands::ActiveMQMessageTemplate, 394
- activemq::wireformat::openwire::utils::MessagePropertyInterce
 - 2562
- cms::Message, 2392
- setFlush
 - activemq::commands::ConsumerControl, 1305
- setFormatId
 - activemq::commands::XATransactionId, 3768
- setFormatter
 - decaf::util::logging::Handler, 1855
- setFragment
 - activemq::util::CompositeData, 1131
 - decaf::internal::net::URIType, 3695
- setGlobalTransactionId
 - activemq::commands::XATransactionId, 3768
- setGroupID
 - activemq::commands::Message, 2371
- setGroupSequence
 - activemq::commands::Message, 2371
- setHost
 - activemq::util::CompositeData, 1131
 - decaf::internal::net::URIType, 3695
- setInitialDelayTime
 - activemq::transport::inactivity::InactivityMonitor, 1877
- setInitialized
 - activemq::transport::failover::FailoverTransport, 1761
- setInitialReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1761
- setInitialRedeliveryDelay
 - activemq::core::policies::DefaultRedeliveryPolicy, 1573
 - activemq::core::RedeliveryPolicy, 2976
- setInput
 - decaf::util::zip::Deflater, 1601, 1602
 - decaf::util::zip::Inflater, 1900, 1901
- setInputStream
 - activemq::transport::IOTransport, 2010
- setInt
 - activemq::commands::ActiveMQMapMessage, 327
 - activemq::util::PrimitiveList, 2796
 - activemq::util::PrimitiveMap, 2806
 - activemq::util::PrimitiveValueNode, 2829
 - cms::MapMessage, 2326
- setIntProperty
 - activemq::commands::ActiveMQMessageTemplate, 395
- activemq::wireformat::openwire::utils::MessagePropertyInterce
 - 2562

- cms::Message, 2392
- setKeepAlive
 - decaf::net::Socket, 3293
- setKeepAliveResponseRequired
 - activemq::transport::inactivity::InactivityMonitor, 1877
- setLastDeliveredSequenceId
 - activemq::commands::RemoveInfo, 2990
 - activemq::core::ActiveMQConsumer, 276
 - activemq::core::ActiveMQSession, 480
- setLastMessageId
 - activemq::commands::MessageAck, 2398
- setLastNakNumber
 - activemq::commands::ReplayCommand, 3045
- setLevel
 - decaf::util::logging::Handler, 1855
 - decaf::util::logging::Logger, 2247
 - decaf::util::logging::LogRecord, 2264
 - decaf::util::zip::Deflater, 1603
- setLimit
 - activemq::util::MemoryUsage, 2357
- setList
 - activemq::util::PrimitiveValueNode, 2829
- setLoggerName
 - decaf::util::logging::LogRecord, 2264
- setLong
 - activemq::commands::ActiveMQMapMessage, 327
 - activemq::util::PrimitiveList, 2797
 - activemq::util::PrimitiveMap, 2806
 - activemq::util::PrimitiveValueNode, 2829
 - cms::MapMessage, 2326
- setLongProperty
 - activemq::commands::ActiveMQMessageTemplate, 395
 - activemq::wireformat::openwire::utils::MessageProperty, 2562
 - cms::Message, 2392
- setMagic
 - activemq::commands::WireFormatInfo, 3725
- setManageable
 - activemq::commands::ConnectionInfo, 1262
- setManaged
 - decaf::internal::util::GenericResource, 1848
- setMap
 - activemq::util::PrimitiveValueNode, 2829
- setMark
 - cms::CMSException, 1077
 - decaf::lang::Exception, 1717
 - decaf::lang::Throwable, 3540
- setMarshaledForm
 - activemq::commands::BaseDataStructure, 767
 - activemq::wireformat::MarshalAware, 2330
- setMarshaledProperties
 - activemq::commands::Message, 2371
 - activemq::commands::WireFormatInfo, 3725
- setMasterBroker
 - activemq::commands::BrokerInfo, 829
- setMaxCacheSize
 - activemq::state::ConnectionStateTracker, 1300
 - activemq::transport::failover::FailoverTransport, 1762
- setMaximumPendingMessageLimit
 - activemq::commands::ConsumerInfo, 1363
- setMaximumRedeliveries
 - activemq::core::policies::DefaultRedeliveryPolicy, 1573
 - activemq::core::RedeliveryPolicy, 2976
- setMaxInactivityDuration
 - activemq::commands::WireFormatInfo, 3725
 - activemq::wireformat::openwire::OpenWireFormat, 2708
- setMaxInactivityDurationInitialDelay
 - activemq::commands::WireFormatInfo, 3725
- setMaxInactivityDurationInitialDelay
 - activemq::wireformat::openwire::OpenWireFormat, 2708
- setMaxReconnectAttempts
 - activemq::transport::failover::FailoverTransport, 1762
- setMaxReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1762
- setMessage
 - activemq::commands::BrokerError, 796
 - activemq::commands::JournalTrace, 2071
 - activemq::commands::MessageDispatch, 2430
 - decaf::lang::Exception, 1718
 - decaf::util::logging::LogRecord, 2264
- setMessageAck
 - activemq::commands::JournalQueueAck, 2016
- setMessageCount
 - activemq::commands::MessageAck, 2398
- setMessageId
 - activemq::commands::JournalTopicAck, 2044

- activemq::commands::Message, 2371
- activemq::commands::MessageDispatchNotification, 2465
- activemq::commands::MessagePull, 2567
- setMessageIdEnabled
 - activemq::cmsutil::CmsTemplate, 1094
- setMessageListener
 - activemq::cmsutil::CachedConsumer, 996
 - activemq::core::ActiveMQConsumer, 276
 - cms::MessageConsumer, 2425
- setMessageSequenceId
 - activemq::commands::JournalTopicAck, 2044
- setMessageTimestampEnabled
 - activemq::cmsutil::CmsTemplate, 1095
- setMimeType
 - activemq::commands::ActiveMQBlobMessage, 169
- setName
 - activemq::commands::ActiveMQBlobMessage, 170
 - decaf::lang::Thread, 3527
- setNeedClientAuth
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2663
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2668
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2683
 - decaf::net::ssl::SSLParameters, 3328
 - decaf::net::ssl::SSLServerSocket, 3334
 - decaf::net::ssl::SSLSocket, 3343
- setNetworkBrokerId
 - activemq::commands::NetworkBridgeFilter, 2616
- setNetworkConnection
 - activemq::commands::BrokerInfo, 829
- setNetworkConsumerPath
 - activemq::commands::ConsumerInfo, 1363
- setNetworkProperties
 - activemq::commands::BrokerInfo, 829
- setNetworkSubscription
 - activemq::commands::ConsumerInfo, 1363
- setNetworkTTL
 - activemq::commands::NetworkBridgeFilter, 2616
- setNoLocal
 - activemq::cmsutil::CmsTemplate, 1095
 - activemq::commands::ConsumerInfo, 1363
- setNoRangeAcks
 - activemq::commands::ConsumerInfo, 1363
- setNumReceivedMessageBeforeFail
 - activemq::transport::mock::MockTransport, 2600
- setNumReceivedMessages
 - activemq::transport::mock::MockTransport, 2600
- setNumSentKeepAlives
 - activemq::transport::mock::MockTransport, 2600
- setNumSentKeepAlivesBeforeFail
 - activemq::transport::mock::MockTransport, 2600
- setNumSentMessageBeforeFail
 - activemq::transport::mock::MockTransport, 2600
- setNumSentMessages
 - activemq::transport::mock::MockTransport, 2600
- setObjectId
 - activemq::commands::RemoveInfo, 2990
- setOOBInline
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2684
 - decaf::net::Socket, 3293
- setOpaque
 - decaf::internal::net::URIType, 3695
- setOperationType
 - activemq::commands::DestinationInfo, 1617
- setOptimizedAcknowledge
 - activemq::commands::ConsumerInfo, 1363
- setOption
 - decaf::internal::net::tcp::TcpSocket, 3504
 - decaf::net::SocketImpl, 3312
- setOrdered
 - activemq::commands::ActiveMQDestination, 287
- setOrderedTarget
 - activemq::commands::ActiveMQDestination, 287
- setOriginalDestination
 - activemq::commands::Message, 2371
- setOriginalTransactionId
 - activemq::commands::Message, 2371
- setOutputStream
 - decaf::util::logging::StreamHandler, 3415
- setOutgoingListener
 - activemq::transport::mock::MockTransport, 2600
- setOutputStream
 - activemq::transport::IOTransport, 2010
- setParameters
 - activemq::util::CompositeData, 1131
- setParent
 - decaf::util::logging::Logger, 2248
- setPassword

- activemq::commands::ConnectionInfo, 1262
- activemq::core::ActiveMQConnection, 248
- activemq::core::ActiveMQConnectionFactory, 260
- setPath
 - activemq::util::CompositeData, 1131
 - decaf::internal::net::URIType, 3695
- setPeerBrokerInfos
 - activemq::commands::BrokerInfo, 829
- setPersistent
 - activemq::commands::Message, 2371
- setPhysicalName
 - activemq::commands::ActiveMQDestination, 287
- setPooledThreadListener
 - decaf::util::concurrent::PooledThread, 2778
- setPort
 - decaf::internal::net::URIType, 3696
- setPreferedWireFormatInfo
 - activemq::wireformat::openwire::OpenWireFormat, 2709
- setPrefetch
 - activemq::commands::ConsumerControl, 1305
- setPrefetchPolicy
 - activemq::core::ActiveMQConnection, 248
 - activemq::core::ActiveMQConnectionFactory, 260
- setPrefetchSize
 - activemq::commands::ConsumerInfo, 1363
- setPrepared
 - activemq::state::TransactionState, 3624
- setPreparedResult
 - activemq::state::TransactionState, 3624
- setPriority
 - activemq::cmsutil::CachedProducer, 1002
 - activemq::cmsutil::CmsTemplate, 1095
 - activemq::commands::ConsumerInfo, 1363
 - activemq::commands::Message, 2371
 - activemq::core::ActiveMQProducer, 430
 - cms::MessageProducer, 2556
 - decaf::lang::Thread, 3527
- setProducerId
 - activemq::commands::Message, 2371
 - activemq::commands::MessageId, 2498
 - activemq::commands::ProducerAck, 2841
 - activemq::commands::ProducerInfo, 2901
- setProducerSequenceId
 - activemq::commands::MessageId, 2498
- setProducerSessionKey
 - activemq::commands::ProducerId, 2873
- setProducerWindowSize
 - activemq::core::ActiveMQConnection, 249
- activemq::core::ActiveMQConnectionFactory, 260
- setProperties
 - activemq::commands::WireFormatInfo, 3726
 - activemq::util::ActiveMQProperties, 434
 - decaf::util::logging::LogManager, 2260
- setProperty
 - activemq::util::ActiveMQProperties, 434
 - activemq::wireformat::stomp::StompFrame, 3402
 - cms::CMSProperties, 1081
 - decaf::lang::System, 3493
 - decaf::util::Properties, 2933
- setProtocols
 - decaf::net::ssl::SSLParameters, 3329
- setPubSubDomain
 - activemq::cmsutil::CmsDestinationAccessor, 1074
 - activemq::cmsutil::CmsTemplate, 1095
- setQuery
 - decaf::internal::net::URIType, 3696
- setQueueBrowserPrefetch
 - activemq::core::policies::DefaultPrefetchPolicy, 1568
 - activemq::core::PrefetchPolicy, 2786
- setQueuePrefetch
 - activemq::core::policies::DefaultPrefetchPolicy, 1568
 - activemq::core::PrefetchPolicy, 2787
- setRandomize
 - activemq::transport::failover::FailoverTransport, 1762
 - activemq::transport::failover::URIPool, 3683
- setReadCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1877
- setReadOnly
 - decaf::internal::nio::ByteBuffer, 943
 - decaf::internal::nio::CharArrayBuffer, 1036
 - decaf::internal::nio::DoubleArrayBuffer, 1692
 - decaf::internal::nio::FloatArrayBuffer, 1799
 - decaf::internal::nio::IntArrayBuffer, 1930
 - decaf::internal::nio::LongArrayBuffer, 2290
 - decaf::internal::nio::ShortArrayBuffer, 3238
- setReadOnlyBody
 - activemq::commands::Message, 2371
- setReadOnlyProperties
 - activemq::commands::Message, 2372
- setRebalanceConnection

- activemq::commands::ConnectionControl, 1175
- setReceiveBufferSize
 - decaf::net::ServerSocket, 3144
 - decaf::net::Socket, 3294
- setReceiveTimeout
 - activemq::cmsutil::CmsTemplate, 1095
- setRecievedByDFBridge
 - activemq::commands::Message, 2372
- setReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1762
- setReconnectTo
 - activemq::commands::ConnectionControl, 1175
- setRedeliveryCounter
 - activemq::commands::Message, 2372
 - activemq::commands::MessageDispatch, 2430
- setRedeliveryPolicy
 - activemq::core::ActiveMQConnection, 249
 - activemq::core::ActiveMQConnectionFactory, 260
 - activemq::core::ActiveMQConsumer, 277
- setRemoteBlobUrl
 - activemq::commands::ActiveMQBlobMessage, 170
- setReplyTo
 - activemq::commands::Message, 2372
- setResponse
 - activemq::transport::correlator::FutureResponse, 1844
- setResponseBuilder
 - activemq::transport::mock::InternalCommandListener, 1987
 - activemq::transport::mock::MockTransport, 2600
- setResponseRequired
 - activemq::commands::BaseCommand, 700
 - activemq::commands::Command, 1111
- setRestoreConsumers
 - activemq::state::ConnectionStateTracker, 1300
- setRestoreProducers
 - activemq::state::ConnectionStateTracker, 1300
- setRestoreSessions
 - activemq::state::ConnectionStateTracker, 1300
- setRestoreTransaction
 - activemq::state::ConnectionStateTracker, 1300
- setResult
 - activemq::commands::IntegerResponse, 1958
- setResume
 - activemq::commands::ConnectionControl, 1175
- setRetroactive
 - activemq::commands::ConsumerInfo, 1363
- setReuseAddress
 - decaf::net::ServerSocket, 3144
 - decaf::net::Socket, 3294
- setScheduledTime
 - decaf::util::TimerTask, 3556
- setScheme
 - activemq::util::CompositeData, 1131
 - decaf::internal::net::URIType, 3696
- setSchemeSpecificPart
 - decaf::internal::net::URIType, 3696
- setSeed
 - decaf::security::SecureRandom, 3120
 - decaf::util::Random, 2957
- setSelector
 - activemq::commands::ConsumerInfo, 1363
 - activemq::commands::SubscriptionInfo, 3437
- setSendBufferSize
 - decaf::net::Socket, 3294
- setSendTimeout
 - activemq::core::ActiveMQConnection, 249
 - activemq::core::ActiveMQConnectionFactory, 261
 - activemq::core::ActiveMQProducer, 430
- setServerAuthority
 - decaf::internal::net::URIType, 3696
- setServiceName
 - activemq::commands::DiscoveryEvent, 1646
- setSessionAcknowledgeMode
 - activemq::cmsutil::CmsAccessor, 1071
- setSessionId
 - activemq::commands::ConsumerId, 1333
 - activemq::commands::ProducerId, 2873
 - activemq::commands::SessionInfo, 3191
- setShort
 - activemq::commands::ActiveMQMapMessage, 327
 - activemq::util::PrimitiveList, 2797
 - activemq::util::PrimitiveMap, 2807
 - activemq::util::PrimitiveValueNode, 2830
 - cms::MapMessage, 2326
- setShortProperty
 - activemq::commands::ActiveMQMessageTemplate, 395
 - activemq::wireformat::openwire::utils::MessagePropertyInterce, 2563

- cms::Message, 2393
- setSize
 - activemq::commands::ProducerAck, 2842
- setSizePrefixDisabled
 - activemq::commands::WireFormatInfo, 3726
 - activemq::wireformat::openwire::OpenWireFormatInfo, 2709
- setSlaveBroker
 - activemq::commands::BrokerInfo, 829
- setSocketImplFactory
 - decaf::net::ServerSocket, 3144
 - decaf::net::Socket, 3295
- setSoLinger
 - decaf::net::Socket, 3295
- setSoTimeout
 - decaf::net::ServerSocket, 3144
 - decaf::net::Socket, 3295
- setSource
 - decaf::internal::net::URIType, 3697
- setSourceFile
 - decaf::util::logging::LogRecord, 2264
- setSourceFunction
 - decaf::util::logging::LogRecord, 2264
- setSourceLine
 - decaf::util::logging::LogRecord, 2265
- setSSLParameters
 - decaf::net::ssl::SSLSocket, 3343
- setStackTrace
 - decaf::lang::Exception, 1718
- setStackTraceElements
 - activemq::commands::BrokerError, 796
- setStackTraceEnabled
 - activemq::commands::WireFormatInfo, 3726
 - activemq::wireformat::openwire::OpenWireFormatInfo, 2709
- setStart
 - activemq::commands::ConsumerControl, 1305
- setStartupMaxReconnectAttempts
 - activemq::transport::failover::FailoverTransport, 1762
- setStop
 - activemq::commands::ConsumerControl, 1305
- setStrategy
 - decaf::util::zip::Deflater, 1603
- setString
 - activemq::commands::ActiveMQMapMessage, 327
 - activemq::util::PrimitiveList, 2797
 - activemq::util::PrimitiveMap, 2807
 - activemq::util::PrimitiveValueNode, 2830
- cms::MapMessage, 2327
- setStringProperty
 - activemq::commands::ActiveMQMessageTemplate, 396
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2563
- cms::Message, 2393
- setSubscriptionName
 - activemq::commands::RemoveSubscriptionInfo, 3018
 - activemq::commands::SubscriptionInfo, 3437
- setSubscribedDestination
 - activemq::commands::SubscriptionInfo, 3437
- setSubscriptionName
 - activemq::commands::ConsumerInfo, 1363
- setSubscriptionName
 - activemq::commands::JournalTopicAck, 2044
- setSuspend
 - activemq::commands::ConnectionControl, 1175
- setSynchronizationRegistered
 - activemq::core::ActiveMQConsumer, 277
- setTargetConsumerId
 - activemq::commands::Message, 2372
- setTcpNoDelay
 - decaf::net::Socket, 3296
- setTcpNoDelayEnabled
 - activemq::commands::WireFormatInfo, 3726
 - activemq::wireformat::openwire::OpenWireFormatInfo, 2709
- setText
 - activemq::commands::ActiveMQTextMessage, 608
 - cms::TextMessage, 3520
- setTextView
 - activemq::commands::MessageId, 2498
- setThrown
 - decaf::util::logging::LogRecord, 2265
- setTightEncodingEnabled
 - activemq::commands::WireFormatInfo, 3726
 - activemq::wireformat::openwire::OpenWireFormatInfo, 2709
- setTime
 - decaf::util::Date, 1562
- setTimeout
 - activemq::commands::DestinationInfo, 1617
 - activemq::commands::MessagePull, 2567

- activemq::transport::failover::FailoverTransport, 1762
- setTimestamp
 - activemq::commands::Message, 2372
 - decaf::util::logging::LogRecord, 2265
- setTimeToLive
 - activemq::cmsutil::CachedProducer, 1003
 - activemq::cmsutil::CmsTemplate, 1095
 - activemq::core::ActiveMQProducer, 430
 - cms::MessageProducer, 2557
- setTopicPrefetch
 - activemq::core::policies::DefaultPrefetchPolicy, 1569
 - activemq::core::PrefetchPolicy, 2787
- setTrackMessages
 - activemq::state::ConnectionStateTracker, 1300
 - activemq::transport::failover::FailoverTransport, 1762
- setTrackTransactionProducers
 - activemq::state::ConnectionStateTracker, 1300
 - activemq::transport::failover::FailoverTransport, 1762
- setTrackTransactions
 - activemq::state::ConnectionStateTracker, 1300
- setTrafficClass
 - decaf::net::Socket, 3296
- setTransactionId
 - activemq::commands::JournalTopicAck, 2044
 - activemq::commands::JournalTransaction, 2098
 - activemq::commands::Message, 2372
 - activemq::commands::MessageAck, 2398
 - activemq::commands::TransactionInfo, 3597
- setTransactionState
 - activemq::state::ProducerState, 2926
- setTransport
 - activemq::transport::failover::BackupTransport, 691
 - activemq::transport::mock::InternalCommandListener, 1987
- setTransportInterruptProcessingComplete
 - activemq::core::ActiveMQConnection, 249
- setTransportListener
 - activemq::transport::failover::FailoverTransport, 1762
 - activemq::transport::IOTransport, 2010
 - activemq::transport::mock::MockTransport, 2601
 - activemq::transport::Transport, 3633
- activemq::transport::TransportFilter, 3642
- setTreadId
 - decaf::util::logging::LogRecord, 2265
- setType
 - activemq::commands::JournalTransaction, 2098
 - activemq::commands::Message, 2372
 - activemq::commands::TransactionInfo, 3597
- setUncaughtExceptionHandler
 - decaf::lang::Thread, 3527
- setupSocketImpl
 - decaf::net::ServerSocket, 3145
- setUri
 - activemq::transport::failover::BackupTransport, 691
- setUsage
 - activemq::util::MemoryUsage, 2357
- setUseAsyncSend
 - activemq::core::ActiveMQConnection, 249
 - activemq::core::ActiveMQConnectionFactory, 261
- setUseClientMode
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2663
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2684
 - decaf::net::ssl::SSLSocket, 3343
- setUseCollisionAvoidance
 - activemq::core::policies::DefaultRedeliveryPolicy, 1573
 - activemq::core::RedeliveryPolicy, 2977
- setUseCompression
 - activemq::core::ActiveMQConnection, 250
 - activemq::core::ActiveMQConnectionFactory, 261
- setUseExponentialBackOff
 - activemq::core::policies::DefaultRedeliveryPolicy, 1573
 - activemq::core::RedeliveryPolicy, 2977
 - activemq::transport::failover::FailoverTransport, 1762
- setUseParentHandlers
 - decaf::util::logging::Logger, 2248
- setUserID
 - activemq::commands::Message, 2372
- setUserInfo
 - decaf::internal::net::URIType, 3697
- setUserName
 - activemq::commands::ConnectionInfo, 1262
- setUsername
 - activemq::core::ActiveMQConnection, 250

- activemq::core::ActiveMQConnectionFactory, 261
- setValid
 - decaf::internal::net::URIType, 3697
- setValue
 - activemq::commands::BrokerId, 800
 - activemq::commands::ConnectionId, 1233
 - activemq::commands::ConsumerId, 1333
 - activemq::commands::LocalTransactionId, 2203
 - activemq::commands::MessageId, 2498
 - activemq::commands::ProducerId, 2873
 - activemq::commands::SessionId, 3164
 - activemq::util::PrimitiveValueNode, 2830
 - decaf::util::Map::Entry, 1707
- setVersion
 - activemq::commands::WireFormatInfo, 3727
 - activemq::wireformat::openwire::OpenWireFormat, 2710
 - activemq::wireformat::stomp::StompWireFormat, 3409
 - activemq::wireformat::WireFormat, 3715
- setWantClientAuth
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 1134
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 1565
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2684
 - decaf::net::ssl::SSLParameters, 3329
 - decaf::net::ssl::SSLServerSocket, 3334
 - decaf::net::ssl::SSLSocket, 3344
- setWasPrepared
 - activemq::commands::JournalTransaction, 2098
- setWindowSize
 - activemq::commands::ProducerInfo, 2901
- setWireFormat
 - activemq::transport::failover::FailoverTransport, 1763
 - activemq::transport::IOTransport, 2011
 - activemq::transport::mock::MockTransport, 2601
 - activemq::transport::Transport, 3633
 - activemq::transport::TransportFilter, 3642
- setWriteCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1877
- SEVERE
 - decaf::util::logging::Level, 2190
- severe
 - decaf::util::logging::Logger, 2248
- Short
 - decaf::lang::Short, 3222
- SHORT_TYPE
 - activemq::util::PrimitiveValueNode, 2821
- ShortArrayBuffer
 - decaf::internal::nio::ShortArrayBuffer, 3232, 3233
- ShortBuffer
 - decaf::nio::ShortBuffer, 3241
- shortValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2815
 - decaf::lang::Byte, 891
 - decaf::lang::Character, 1026
 - decaf::lang::Double, 1680
 - decaf::lang::Float, 1788
 - decaf::lang::Integer, 1952
 - decaf::lang::Long, 2278
 - decaf::lang::Number, 2655
 - decaf::lang::Short, 3227
- shutdown
 - activemq::state::ConnectionState, 1293
 - activemq::state::SessionState, 3219
 - activemq::state::TransactionState, 3624
 - activemq::threads::CompositeTaskRunner, 1134
 - activemq::threads::DedicatedTaskRunner, 1565
 - activemq::threads::TaskRunner, 3497
- ShutdownInfo
 - activemq::commands::ShutdownInfo, 3250
- ShutdownInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 3261
 - activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 3257
 - activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 3269
 - activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 3273
 - activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 3265
 - activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3253
- shutdownInput
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2685
 - decaf::internal::net::tcp::TcpSocket, 3505
 - decaf::net::Socket, 3296
 - decaf::net::SocketImpl, 3312
- shutdownLibrary
 - activemq::library::ActiveMQCPP, 279
- shutdownNetworking
 - decaf::internal::net::Network, 2613
- shutdownOutput

- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2685
- decaf::internal::net::tcp::TcpSocket, 3505
- decaf::net::Socket, 3296
- decaf::net::SocketImpl, 3312
- shutdownRuntime
 - decaf::lang::Runtime, 3113
- signal
 - decaf::util::concurrent::locks::Condition, 1162
- signalAll
 - decaf::util::concurrent::locks::Condition, 1162
- SignatureException
 - decaf::security::SignatureException, 3276, 3277
- signum
 - decaf::lang::Integer, 1952
 - decaf::lang::Long, 2278
 - decaf::lang::Math, 2349, 2350
- SimpleFormatter
 - decaf::util::logging::SimpleFormatter, 3279
- SimpleLogger
 - decaf::util::logging::SimpleLogger, 3280
- SIZE
 - decaf::lang::Byte, 893
 - decaf::lang::Character, 1026
 - decaf::lang::Double, 1683
 - decaf::lang::Float, 1791
 - decaf::lang::Integer, 1956
 - decaf::lang::Long, 2281
 - decaf::lang::Short, 3229
- size
 - activemq::commands::ProducerAck, 2842
 - activemq::core::MessageDispatchChannel, 2435
 - activemq::wireformat::openwire::utils::HexTable, 1858
 - decaf::internal::util::TimerTaskHeap, 3558
 - decaf::io::ByteArrayOutputStream, 953
 - decaf::io::DataOutputStream, 1475
 - decaf::util::Collection, 1106
 - decaf::util::concurrent::ConcurrentStlMap, 1153
 - decaf::util::concurrent::SynchronousQueue, 3486
 - decaf::util::Map, 2316
 - decaf::util::PriorityQueue, 2839
 - decaf::util::Properties, 2934
 - decaf::util::StlList, 3369
 - decaf::util::StlMap, 3379
 - decaf::util::StlQueue, 3387
 - decaf::util::StlSet, 3395
 - gz_state, 1851
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2698
- decaf::internal::net::tcp::TcpSocketInputStream, 3508
- decaf::io::BlockingByteArrayInputStream, 774
- decaf::io::BufferedInputStream, 866
- decaf::io::ByteArrayInputStream, 950
- decaf::io::FilterInputStream, 1776
- decaf::io::InputStream, 1916
- decaf::io::PushbackInputStream, 2945
- decaf::io::Reader, 2966
- decaf::util::zip::CheckedInputStream, 1057
- decaf::util::zip::InflaterInputStream, 1908
- gz_state, 1851
- skipBytes
 - decaf::io::DataInput, 1459
 - decaf::io::DataInputStream, 1468
- slaveBroker
 - activemq::commands::BrokerInfo, 831
- sleep
 - decaf::lang::Thread, 3528
 - decaf::util::concurrent::TimeUnit, 3563
- SLEEPING
 - decaf::lang::Thread, 3524
- slice
 - decaf::internal::nio::ByteBuffer, 943
 - decaf::internal::nio::CharArrayBuffer, 1036
 - decaf::internal::nio::DoubleArrayBuffer, 1692
 - decaf::internal::nio::FloatArrayBuffer, 1799
 - decaf::internal::nio::IntArrayBuffer, 1930
 - decaf::internal::nio::LongArrayBuffer, 2290
 - decaf::internal::nio::ShortArrayBuffer, 3238
 - decaf::nio::ByteBuffer, 977
 - decaf::nio::CharBuffer, 1051
 - decaf::nio::DoubleBuffer, 1702
 - decaf::nio::FloatBuffer, 1809
 - decaf::nio::IntBuffer, 1940
 - decaf::nio::LongBuffer, 2300
 - decaf::nio::ShortBuffer, 3248
- Socket
 - decaf::net::Socket, 3285, 3286
- SOCKET_OPTION_BINDADDR
 - decaf::net::SocketOptions, 3316
- SOCKET_OPTION_BROADCAST
 - decaf::net::SocketOptions, 3316
- SOCKET_OPTION_IP_MULTICAST_IF
 - decaf::net::SocketOptions, 3316
- SOCKET_OPTION_IP_MULTICAST_IF2
 - decaf::net::SocketOptions, 3316

- SOCKET_OPTION_IP_MULTICAST_LOOP
 - decaf::net::SocketOptions, 3317
- SOCKET_OPTION_IP_TOS
 - decaf::net::SocketOptions, 3317
- SOCKET_OPTION_KEEPAIVE
 - decaf::net::SocketOptions, 3317
- SOCKET_OPTION_LINGER
 - decaf::net::SocketOptions, 3317
- SOCKET_OPTION_OOBLINE
 - decaf::net::SocketOptions, 3317
- SOCKET_OPTION_RCVBUF
 - decaf::net::SocketOptions, 3318
- SOCKET_OPTION_REUSEADDR
 - decaf::net::SocketOptions, 3318
- SOCKET_OPTION_SNDBUF
 - decaf::net::SocketOptions, 3318
- SOCKET_OPTION_TCP_NODELAY
 - decaf::net::SocketOptions, 3318
- SOCKET_OPTION_TIMEOUT
 - decaf::net::SocketOptions, 3318
- SocketException
 - decaf::net::SocketException, 3299, 3300
- SocketFactory
 - decaf::net::SocketFactory, 3302
- SocketFileDescriptor
 - decaf::internal::net::SocketFileDescriptor, 3305
- SocketImpl
 - decaf::net::SocketImpl, 3308
- SocketTimeoutException
 - decaf::net::SocketTimeoutException, 3319, 3320
- sqrt
 - decaf::lang::Math, 2351
- src/main/activemq/cmsutil/CachedConsumer.h, 3799
- src/main/activemq/cmsutil/CachedProducer.h, 3799
- src/main/activemq/cmsutil/CmsAccessor.h, 3800
- src/main/activemq/cmsutil/CmsDestinationAccessor.h, 3800
- src/main/activemq/cmsutil/CmsTemplate.h, 3801
- src/main/activemq/cmsutil/DestinationResolver.h, 3801
- src/main/activemq/cmsutil/DynamicDestinationResolver.h, 3802
- src/main/activemq/cmsutil/MessageCreator.h, 3802
- src/main/activemq/cmsutil/PooledSession.h, 3803
- src/main/activemq/cmsutil/ProducerCallback.h, 3803
- src/main/activemq/cmsutil/ResourceLifecycleManager.h, 3804
- src/main/activemq/cmsutil/SessionCallback.h, 3805
- src/main/activemq/cmsutil/SessionPool.h, 3805
- src/main/activemq/commands/ActiveMQBlobMessage.h, 3806
- src/main/activemq/commands/ActiveMQBytesMessage.h, 3806
- src/main/activemq/commands/ActiveMQDestination.h, 3807
- src/main/activemq/commands/ActiveMQMapMessage.h, 3808
- src/main/activemq/commands/ActiveMQMessage.h, 3808
- src/main/activemq/commands/ActiveMQMessageTemplate.h, 3809
- src/main/activemq/commands/ActiveMQObjectMessage.h, 3809
- src/main/activemq/commands/ActiveMQQueue.h, 3810
- src/main/activemq/commands/ActiveMQStreamMessage.h, 3810
- src/main/activemq/commands/ActiveMQTempDestination.h, 3811
- src/main/activemq/commands/ActiveMQTempQueue.h, 3812
- src/main/activemq/commands/ActiveMQTempTopic.h, 3812
- src/main/activemq/commands/ActiveMQTextMessage.h, 3813
- src/main/activemq/commands/ActiveMQTopic.h, 3813
- src/main/activemq/commands/BaseCommand.h, 3814
- src/main/activemq/commands/BaseDataStructure.h, 3814
- src/main/activemq/commands/BooleanExpression.h, 3815
- src/main/activemq/commands/BrokerError.h, 3815
- src/main/activemq/commands/BrokerId.h, 3816
- src/main/activemq/commands/BrokerInfo.h, 3816
- src/main/activemq/commands/Command.h, 3817
- src/main/activemq/commands/ConnectionControl.h, 3817
- src/main/activemq/commands/ConnectionError.h, 3818

src/main/activemq/commands/ConnectionId.h,	src/main/activemq/commands/NetworkBridgeFilter.h,
3818	3834
src/main/activemq/commands/ConnectionInfo.h,	src/main/activemq/commands/PartialCommand.h,
3819	3834
src/main/activemq/commands/ConsumerControl.h,	src/main/activemq/commands/ProducerAck.h,
3819	3835
src/main/activemq/commands/ConsumerId.h,	src/main/activemq/commands/ProducerId.h,
3820	3835
src/main/activemq/commands/ConsumerInfo.h,	src/main/activemq/commands/ProducerInfo.h,
3821	3836
src/main/activemq/commands/ControlCommands.h,	src/main/activemq/commands/RemoveInfo.h,
3821	3836
src/main/activemq/commands/DataArrayResponse.h,	src/main/activemq/commands/RemoveSubscriptionInfo.h,
3822	3837
src/main/activemq/commands/DataResponse.h,	src/main/activemq/commands/ReplayCommand.h,
3822	3837
src/main/activemq/commands/DataStructure.h,	src/main/activemq/commands/Response.h,
3823	3838
src/main/activemq/commands/DestinationInfo.h,	src/main/activemq/commands/SessionId.h,
3823	3838
src/main/activemq/commands/DiscoveryEvent.h,	src/main/activemq/commands/SessionInfo.h,
3824	3839
src/main/activemq/commands/ExceptionResponse.h,	src/main/activemq/commands/ShutdownInfo.h,
3824	3839
src/main/activemq/commands/FlushCommand.h,	src/main/activemq/commands/SubscriptionInfo.h,
3825	3840
src/main/activemq/commands/IntegerResponse.h,	src/main/activemq/commands/TransactionId.h,
3825	3840
src/main/activemq/commands/JournalQueueAck.h,	src/main/activemq/commands/TransactionInfo.h,
3826	3841
src/main/activemq/commands/JournalTopicAck.h,	src/main/activemq/commands/WireFormatInfo.h,
3826	3841
src/main/activemq/commands/JournalTrace.h,	src/main/activemq/commands/XATransactionId.h,
3827	3842
src/main/activemq/commands/JournalTransaction.h,	src/main/activemq/core/ActiveMQAckHandler.h,
3827	3842
src/main/activemq/commands/KeepAliveInfo.h,	src/main/activemq/core/ActiveMQConnection.h,
3828	3843
src/main/activemq/commands/LastPartialCommand.h,	src/main/activemq/core/ActiveMQConnectionFactory.h,
3828	3844
src/main/activemq/commands/LocalTransactionId.h,	src/main/activemq/core/ActiveMQConnectionMetaData.h,
3829	3844
src/main/activemq/commands/Message.h,	src/main/activemq/core/ActiveMQConstants.h,
3829	3845
src/main/activemq/commands/MessageAck.h,	src/main/activemq/core/ActiveMQConsumer.h,
3831	3845
src/main/activemq/commands/MessageDispatch.h,	src/main/activemq/core/ActiveMQProducer.h,
3831	3846
src/main/activemq/commands/MessageDispatchNotFinal.h,	src/main/activemq/core/ActiveMQQueueBrowser.h,
3832	3847
src/main/activemq/commands/MessageId.h,	src/main/activemq/core/ActiveMQSession.h,
3832	3847
src/main/activemq/commands/MessagePull.h,	src/main/activemq/core/ActiveMQSessionExecutor.h,
3833	3848

src/main/activemq/core/ActiveMQTransactionCoordinator.h, 3849

src/main/activemq/core/DispatchData.h, 3849

src/main/activemq/core/Dispatcher.h, 3850

src/main/activemq/core/MessageDispatchChannel.h, 3850

src/main/activemq/core/policies/DefaultPrefetchPolicy.h, 3851

src/main/activemq/core/policies/DefaultRedeliveryPolicy.h, 3851

src/main/activemq/core/PrefetchPolicy.h, 3852

src/main/activemq/core/RedeliveryPolicy.h, 3852

src/main/activemq/core/Synchronization.h, 3853

src/main/activemq/exceptions/ActiveMQException.h, 3853

src/main/activemq/exceptions/BrokerException.h, 3854

src/main/activemq/exceptions/ExceptionDefines.h, 3854

src/main/activemq/io/LoggingInputStream.h, 3858

src/main/activemq/io/LoggingOutputStream.h, 3859

src/main/activemq/library/ActiveMQCPP.h, 3859

src/main/activemq/state/CommandVisitor.h, 3860

src/main/activemq/state/CommandVisitorAdapter.h, 3860

src/main/activemq/state/ConnectionState.h, 3861

src/main/activemq/state/ConnectionStateTracker.h, 3862

src/main/activemq/state/ConsumerState.h, 3863

src/main/activemq/state/ProducerState.h, 3863

src/main/activemq/state/SessionState.h, 3864

src/main/activemq/state/Tracked.h, 3865

src/main/activemq/state/TransactionState.h, 3865

src/main/activemq/threads/CompositeTask.h, 3866

src/main/activemq/threads/CompositeTaskRunner.h, 3866

src/main/activemq/threads/DedicatedTaskRunner.h, 3867

src/main/activemq/threads/Task.h, 3867

src/main/activemq/threads/TaskRunner.h, 3868

src/main/activemq/transport/AbstractTransportFactory.h, 3868

src/main/activemq/transport/CompositeTransport.h, 3869

src/main/activemq/transport/correlator/FutureResponse.h, 3869

src/main/activemq/transport/correlator/ResponseCorrelator.h, 3870

src/main/activemq/transport/DefaultTransportListener.h, 3871

src/main/activemq/transport/failover/BackupTransport.h, 3871

src/main/activemq/transport/failover/BackupTransportPool.h, 3872

src/main/activemq/transport/failover/CloseTransportsTask.h, 3872

src/main/activemq/transport/failover/FailoverTransport.h, 3873

src/main/activemq/transport/failover/FailoverTransportFactory.h, 3874

src/main/activemq/transport/failover/FailoverTransportListener.h, 3874

src/main/activemq/transport/failover/URIPool.h, 3875

src/main/activemq/transport/inactivity/InactivityMonitor.h, 3875

src/main/activemq/transport/inactivity/ReadChecker.h, 3876

src/main/activemq/transport/inactivity/WriteChecker.h, 3876

src/main/activemq/transport/IOTransport.h, 3877

src/main/activemq/transport/logging/LoggingTransport.h, 3878

src/main/activemq/transport/mock/InternalCommandListener.h, 3878

src/main/activemq/transport/mock/MockTransport.h, 3879

src/main/activemq/transport/mock/MockTransportFactory.h, 3880

src/main/activemq/transport/mock/ResponseBuilder.h, 3880

src/main/activemq/transport/tcp/SslTransport.h, 3881

src/main/activemq/transport/tcp/SslTransportFactory.h, 3881

src/main/activemq/transport/tcp/TcpTransport.h, 3882

src/main/activemq/transport/tcp/TcpTransportFactory.h, 3882

src/main/activemq/transport/Transport.h, 3883

src/main/activemq/transport/TransportFactory.h, 3884

src/main/activemq/transport/TransportFilter.h, 3884

src/main/activemq/transport/TransportListener.h,	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQT
3885	3937
src/main/activemq/transport/TransportRegistry.h,	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQT
3885	3941
src/main/activemq/util/ActiveMQProperties.h,	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQT
3886	3945
src/main/activemq/util/CMSExceptionSupport.h,	src/main/activemq/wireformat/openwire/marshall/v1/BaseComm
3886	3949
src/main/activemq/util/CompositeData.h,	src/main/activemq/wireformat/openwire/marshall/v1/BrokerIdMa
3888	3953
src/main/activemq/util/Config.h, 3889	src/main/activemq/wireformat/openwire/marshall/v1/BrokerInfoM
src/main/activemq/util/IdGenerator.h, 3890	3957
src/main/activemq/util/LongSequenceGenerator.h,	src/main/activemq/wireformat/openwire/marshall/v1/Connection
3890	3961
src/main/activemq/util/MarshallingSupport.h,	src/main/activemq/wireformat/openwire/marshall/v1/Connection
3891	3965
src/main/activemq/util/MemoryUsage.h, 3891	src/main/activemq/wireformat/openwire/marshall/v1/Connection
src/main/activemq/util/PrimitiveList.h, 3892	3969
src/main/activemq/util/PrimitiveMap.h, 3892	src/main/activemq/wireformat/openwire/marshall/v1/Connection
src/main/activemq/util/PrimitiveValueConverter.h,	3973
3893	src/main/activemq/wireformat/openwire/marshall/v1/ConsumerC
src/main/activemq/util/PrimitiveValueNode.h,	3977
3893	src/main/activemq/wireformat/openwire/marshall/v1/ConsumerId
src/main/activemq/util/URISupport.h, 3894	3981
src/main/activemq/util/Usage.h, 3894	src/main/activemq/wireformat/openwire/marshall/v1/ConsumerIn
src/main/activemq/wireformat/MarshalAware.h,	3985
3895	src/main/activemq/wireformat/openwire/marshall/v1/ControlCom
src/main/activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h,	3989
3895	src/main/activemq/wireformat/openwire/marshall/v1/DataArrayF
src/main/activemq/wireformat/openwire/marshall/DataStreamMarshaller.h,	3993
3896	src/main/activemq/wireformat/openwire/marshall/v1/DataRespon
src/main/activemq/wireformat/openwire/marshall/PrimitiveTypesMarshaller.h,	3997
3897	src/main/activemq/wireformat/openwire/marshall/v1/Destination
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQBlobMessageMarshaller.h,	4001
3897	src/main/activemq/wireformat/openwire/marshall/v1/DiscoveryE
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQBytesMessageMarshaller.h,	4005
3901	src/main/activemq/wireformat/openwire/marshall/v1/ExceptionR
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQDestinationMarshaller.h,	4009
3905	src/main/activemq/wireformat/openwire/marshall/v1/FlushComm
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQMapMessageMarshaller.h,	4013
3909	src/main/activemq/wireformat/openwire/marshall/v1/IntegerResp
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQMessageMarshaller.h,	4017
3913	src/main/activemq/wireformat/openwire/marshall/v1/JournalQue
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQObjectMessageMarshaller.h,	4021
3917	src/main/activemq/wireformat/openwire/marshall/v1/JournalTop
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQQueueMarshaller.h,	4025
3921	src/main/activemq/wireformat/openwire/marshall/v1/JournalTrac
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQStreamMessageMarshaller.h,	4029
3925	src/main/activemq/wireformat/openwire/marshall/v1/JournalTran
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempDestinationMarshaller.h,	4033
3929	src/main/activemq/wireformat/openwire/marshall/v1/KeepAliveIn
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempQueueMarshaller.h,	4037
3933	src/main/activemq/wireformat/openwire/marshall/v1/LastPartial
	4041

src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4045	3906
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4049	3910
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4052	3914
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4056	3918
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4060	3922
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4064	3926
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4068	3930
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4072	3934
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4076	3938
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4080	3942
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4084	3946
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4088	3950
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4092	3954
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4096	3958
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4100	3962
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4104	3966
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4108	3970
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4112	3974
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4116	3978
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4120	3982
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4124	3986
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4128	3990
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4132	3994
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4136	3998
src/main/activemq/wireformat/openwire/marshaller/v1/InboundTransactionWireMarshaller/openwire/marshaller/v2/ActiveMQD	
4140	4002
src/main/activemq/wireformat/openwire/marshaller/v2/InboundTransactionWireMarshaller/openwire/marshaller/v2/DiscoveryE	
3898	4006
src/main/activemq/wireformat/openwire/marshaller/v2/InboundTransactionWireMarshaller/openwire/marshaller/v2/ExceptionR	
3902	4010

src/main/activemq/wireformat/openwire/marshaller/v2/FlushContinuumWireMarshaller/openwire/marshaller/v2/ShutDownIn	4014	4121
src/main/activemq/wireformat/openwire/marshaller/v2/IntegerResponseWireMarshaller/openwire/marshaller/v2/Subscription	4018	4125
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveQueueAckWireMarshaller/openwire/marshaller/v2/Transaction	4022	4129
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveQueueAckWireMarshaller/openwire/marshaller/v2/Transaction	4026	4133
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveQueueAckWireMarshaller/openwire/marshaller/v2/WireFormat	4030	4137
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveQueueAckWireMarshaller/openwire/marshaller/v2/XATransact	4034	4141
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQB	4038	3899
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQB	4042	3903
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQD	4046	3907
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQM	4050	3911
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQM	4053	3915
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQC	4057	3919
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQC	4061	3923
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQS	4065	3927
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQT	4069	3931
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQT	4073	3935
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQT	4077	3939
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQT	4081	3943
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ActiveMQT	4085	3947
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/BaseComm	4089	3951
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/BrokerIdMa	4093	3955
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/BrokerInfoM	4097	3959
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/Connection	4101	3963
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/Connection	4105	3967
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/Connection	4109	3971
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/Connection	4113	3975
src/main/activemq/wireformat/openwire/marshaller/v2/JainActiveInfoMarshaller/openwire/marshaller/v3/ConsumerC	4117	3979

src/main/activemq/wireformat/openwire/marshaller/v3/ConsumerIdMarshaller/openwire/marshaller/v3/ProducerIdMarshaller 3983 4090

src/main/activemq/wireformat/openwire/marshaller/v3/ConsumerInfoMarshaller/openwire/marshaller/v3/ProducerInfoMarshaller 3987 4094

src/main/activemq/wireformat/openwire/marshaller/v3/ControlCommandMarshaller/openwire/marshaller/v3/RemoveInfoMarshaller 3991 4098

src/main/activemq/wireformat/openwire/marshaller/v3/DataActiveResponseMarshaller/openwire/marshaller/v3/RemoveSubscriptionMarshaller 3995 4102

src/main/activemq/wireformat/openwire/marshaller/v3/DataResponseMarshaller/openwire/marshaller/v3/ReplayCommandMarshaller 3999 4106

src/main/activemq/wireformat/openwire/marshaller/v3/DeactivationInfoMarshaller/openwire/marshaller/v3/ResponseMarshaller 4003 4110

src/main/activemq/wireformat/openwire/marshaller/v3/DisconnectWireFormatMarshaller/openwire/marshaller/v3/SessionIdMarshaller 4007 4114

src/main/activemq/wireformat/openwire/marshaller/v3/ExceptionResponseMarshaller/openwire/marshaller/v3/SessionInfoMarshaller 4011 4118

src/main/activemq/wireformat/openwire/marshaller/v3/FlushActiveMQWireFormatMarshaller/openwire/marshaller/v3/ShutDownInfoMarshaller 4015 4122

src/main/activemq/wireformat/openwire/marshaller/v3/InformerResponseMarshaller/openwire/marshaller/v3/SubscriptionMarshaller 4019 4126

src/main/activemq/wireformat/openwire/marshaller/v3/JournalQueueWireFormatMarshaller/openwire/marshaller/v3/TransactionMarshaller 4023 4130

src/main/activemq/wireformat/openwire/marshaller/v3/JournalTopicWireFormatMarshaller/openwire/marshaller/v3/TransactionInfoMarshaller 4027 4134

src/main/activemq/wireformat/openwire/marshaller/v3/JournalTransactionMarshaller/openwire/marshaller/v3/WireFormatMarshaller 4031 4138

src/main/activemq/wireformat/openwire/marshaller/v3/JournalTransactionWireFormatMarshaller/openwire/marshaller/v3/XATransactionMarshaller 4035 4142

src/main/activemq/wireformat/openwire/marshaller/v3/LocalActiveMQMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4039 3899

src/main/activemq/wireformat/openwire/marshaller/v3/LocalActiveMQWireFormatMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4043 3903

src/main/activemq/wireformat/openwire/marshaller/v3/LocalTransactionWireFormatMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4047 3907

src/main/activemq/wireformat/openwire/marshaller/v3/MarshallerFactory/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4050 3911

src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveMQMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4053 3915

src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveMQDispatchMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4057 3919

src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveMQDispatchNotificationMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4062 3923

src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveMQWireFormatMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4066 3927

src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveMQWireFormatMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4070 3931

src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveMQWireFormatMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4074 3935

src/main/activemq/wireformat/openwire/marshaller/v3/NewYorkBridgeFilterMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4078 3939

src/main/activemq/wireformat/openwire/marshaller/v3/PartialCommandMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4082 3943

src/main/activemq/wireformat/openwire/marshaller/v3/ProducerIdMarshaller/openwire/marshaller/v4/ActiveMQBrokerMarshaller 4086 3947

- src/main/activemq/wireformat/openwire/marshaller/v4/BaseActiveAndMarshalHeader/openwire/marshaller/v4/MessageDis
3951 4058
- src/main/activemq/wireformat/openwire/marshaller/v4/BrokerIdMarshaller/openwire/marshaller/v4/MessageDis
3955 4062
- src/main/activemq/wireformat/openwire/marshaller/v4/BrokerInfoMarshaller/openwire/marshaller/v4/MessageDis
3959 4066
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveControlMarshaller/openwire/marshaller/v4/MessageMa
3963 4070
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveErrorMarshaller/openwire/marshaller/v4/MessagePul
3967 4074
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveIdMarshaller/openwire/marshaller/v4/NetworkBri
3971 4078
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveInfoMarshaller/openwire/marshaller/v4/PartialCom
3975 4082
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/ProducerAc
3979 4086
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/ProducerId
3983 4090
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/ProducerInf
3987 4094
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/RemoveInfo
3991 4098
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/RemoveSub
3995 4102
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/ReplayCom
3999 4106
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/ResponseM
4003 4110
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/SessionIdM
4007 4114
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/SessionInfo
4011 4118
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/ShutDownIn
4015 4122
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/Subscription
4019 4126
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/Transaction
4023 4130
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/Transaction
4027 4134
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/WireFormat
4031 4138
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v4/XATransact
4035 4142
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v5/ActiveMQB
4039 3900
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v5/ActiveMQB
4043 3904
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v5/ActiveMQD
4047 3908
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v5/ActiveMQM
4051 3912
- src/main/activemq/wireformat/openwire/marshaller/v4/CompactiveQueueMarshaller/openwire/marshaller/v5/ActiveMQM
4054 3916

src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQObjectMessageMarshaller	4028
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQQueueMessageMarshaller	4032
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQSimpleMessageMarshaller	4036
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQTempDestinationMarshaller	4040
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQTempQueueMarshaller	4044
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQTempTopicMarshaller	4048
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQTextMessageMarshaller	4051
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQTopicWireMarshaller	4055
src/main/activemq/wireformat/openwire/marshaller/v5/BaseActiveMQMarshaller	4059
src/main/activemq/wireformat/openwire/marshaller/v5/BrokerIdMarshaller	4063
src/main/activemq/wireformat/openwire/marshaller/v5/BrokerIdFormatMarshaller	4067
src/main/activemq/wireformat/openwire/marshaller/v5/ConnectionControlMarshaller	4071
src/main/activemq/wireformat/openwire/marshaller/v5/ConnectionErrorMarshaller	4075
src/main/activemq/wireformat/openwire/marshaller/v5/ConnectionIdMarshaller	4079
src/main/activemq/wireformat/openwire/marshaller/v5/ConnectionInfoMarshaller	4083
src/main/activemq/wireformat/openwire/marshaller/v5/ConsumerGroupInfoMarshaller	4087
src/main/activemq/wireformat/openwire/marshaller/v5/ConsumerIdMarshaller	4091
src/main/activemq/wireformat/openwire/marshaller/v5/ConsumerInfoMarshaller	4095
src/main/activemq/wireformat/openwire/marshaller/v5/ControlCommandMarshaller	4099
src/main/activemq/wireformat/openwire/marshaller/v5/DataActiveResponseMarshaller	4103
src/main/activemq/wireformat/openwire/marshaller/v5/DataResponseMarshaller	4107
src/main/activemq/wireformat/openwire/marshaller/v5/DestinationInfoMarshaller	4111
src/main/activemq/wireformat/openwire/marshaller/v5/DisconnectErrorMarshaller	4115
src/main/activemq/wireformat/openwire/marshaller/v5/ExceptionResponseMarshaller	4119
src/main/activemq/wireformat/openwire/marshaller/v5/FlushActiveMQWireMarshaller	4123
src/main/activemq/wireformat/openwire/marshaller/v5/IntegerResponseMarshaller	4127
src/main/activemq/wireformat/openwire/marshaller/v5/LocalQueueAckMarshaller	4131

src/main/activemq/wireformat/openwire/marshaller/v5/FrameSectionInfoMarshaller/openwire/marshaller/v6/DataArrayF	3997
4135	
src/main/activemq/wireformat/openwire/marshaller/v5/WireFormatInfoMarshaller/openwire/marshaller/v6/DataRespon	4001
4139	
src/main/activemq/wireformat/openwire/marshaller/v5/WriteActiveQueueIdMarshaller/openwire/marshaller/v6/Destination	4005
4143	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQBlobMessageMarshaller/openwire/marshaller/v6/DiscoveryE	4009
3901	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQBytesMessageMarshaller/openwire/marshaller/v6/ExceptionR	4013
3905	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQDestinationFormatMarshaller/openwire/marshaller/v6/FlushComm	4017
3909	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQMapMessageMarshaller/openwire/marshaller/v6/IntegerResp	4021
3913	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQMessageMarshaller/openwire/marshaller/v6/JournalQue	4025
3917	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQObjectMessageMarshaller/openwire/marshaller/v6/JournalTop	4029
3921	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQQueueFormatMarshaller/openwire/marshaller/v6/JournalTrac	4033
3925	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQSimpleMessageMarshaller/openwire/marshaller/v6/JournalTran	4037
3929	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTopicDestinationMarshaller/openwire/marshaller/v6/KeepAliveIn	4041
3933	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTopicQueueFormatMarshaller/openwire/marshaller/v6/LastPartial	4045
3937	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTopicQueueFormatMarshaller/openwire/marshaller/v6/LocalTrans	4049
3941	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTextMessageMarshaller/openwire/marshaller/v6/MarshallerE	4052
3945	
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTopicInfoMarshaller/openwire/marshaller/v6/MessageAck	4055
3949	
src/main/activemq/wireformat/openwire/marshaller/v6/BaseCommandMarshaller/openwire/marshaller/v6/MessageDis	4059
3953	
src/main/activemq/wireformat/openwire/marshaller/v6/BrokerIdMarshaller/openwire/marshaller/v6/MessageDis	4064
3957	
src/main/activemq/wireformat/openwire/marshaller/v6/BrokerIdMessageFormat/openwire/marshaller/v6/MessageIdM	4068
3961	
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectiveControlMarshaller/openwire/marshaller/v6/MessageMa	4072
3965	
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectiveInfoMarshaller/openwire/marshaller/v6/MessagePul	4076
3969	
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectiveInfoMarshaller/openwire/marshaller/v6/NetworkBri	4080
3973	
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectiveInfoMarshaller/openwire/marshaller/v6/PartialCom	4084
3977	
src/main/activemq/wireformat/openwire/marshaller/v6/ConsumerGroupMarshaller/openwire/marshaller/v6/ProducerAc	4088
3981	
src/main/activemq/wireformat/openwire/marshaller/v6/ConsumerIdMarshaller/openwire/marshaller/v6/ProducerId	4092
3985	
src/main/activemq/wireformat/openwire/marshaller/v6/ConsumerInfoMarshaller/openwire/marshaller/v6/ProducerInf	4096
3989	
src/main/activemq/wireformat/openwire/marshaller/v6/CommandGroupMarshaller/openwire/marshaller/v6/RemoveInfo	4100
3993	

src/main/activemq/wireformat/openwire/marshaller/v6/ResponseMarshaller.h, 4104
 src/main/activemq/wireformat/openwire/marshaller/v6/ResponseMarshaller.h, 4108
 src/main/activemq/wireformat/openwire/marshaller/v6/ResponseMarshaller.h, 4112
 src/main/activemq/wireformat/openwire/marshaller/v6/ResponseMarshaller.h, 4116
 src/main/activemq/wireformat/openwire/marshaller/v6/ResponseMarshaller.h, 4120
 src/main/activemq/wireformat/openwire/marshaller/v6/ResponseMarshaller.h, 4124
 src/main/activemq/wireformat/openwire/marshaller/v6/ResponseMarshaller.h, 4128
 src/main/activemq/wireformat/openwire/marshaller/v6/ResponseMarshaller.h, 4132
 src/main/activemq/wireformat/openwire/marshaller/v6/ResponseMarshaller.h, 4136
 src/main/activemq/wireformat/openwire/marshaller/v6/ResponseMarshaller.h, 4140
 src/main/activemq/wireformat/openwire/marshaller/v6/ResponseMarshaller.h, 4144
 src/main/activemq/wireformat/openwire/OpenWireFormat.h, 4144
 src/main/activemq/wireformat/openwire/OpenWireFormat.h, 4145
 src/main/activemq/wireformat/openwire/OpenWireFormat.h, 4146
 src/main/activemq/wireformat/openwire/OpenWireFormat.h, 4146
 src/main/activemq/wireformat/openwire/Utils/BooleanStream.h, 4147
 src/main/activemq/wireformat/openwire/Utils/HexTable.h, 4147
 src/main/activemq/wireformat/openwire/Utils/MessageProperties.h, 4148
 src/main/activemq/wireformat/stomp/StompCommand.h, 4148
 src/main/activemq/wireformat/stomp/StompFrame.h, 4149
 src/main/activemq/wireformat/stomp/StompHelper.h, 4150
 src/main/activemq/wireformat/stomp/StompWireFormat.h, 4150
 src/main/activemq/wireformat/stomp/StompWireFormat.h, 4151
 src/main/activemq/wireformat/WireFormat.h, 4151
 src/main/activemq/wireformat/WireFormatFactory.h, 4152
 src/main/activemq/wireformat/WireFormatNegotiator.h, 4153
 src/main/activemq/wireformat/WireFormatRegistry.h, 4153
 src/main/cms/Closeable.h, 4154
 src/main/cms/CMSException.h, 4155
 src/main/cms/CMSProperties.h, 4156
 src/main/cms/CMSResponseMarshaller.h, 4156
 src/main/cms/Config.h, 3889
 src/main/cms/ConnectionFactory.h, 4157
 src/main/cms/ConnectionFactory.h, 4157
 src/main/cms/DeliveryMode.h, 4158
 src/main/cms/ExceptionListener.h, 4159
 src/main/cms/InvalidClientIdException.h, 4160
 src/main/cms/InvalidDestinationException.h, 4160
 src/main/cms/InvalidSelectorException.h, 4161
 src/main/cms/InvalidSelectorException.h, 4161
 src/main/cms/Message.h, 3830
 src/main/cms/Message.h, 4162
 src/main/cms/MessageEnumeration.h, 4162
 src/main/cms/MessageEOFException.h, 4163
 src/main/cms/MessageFormatException.h, 4163
 src/main/cms/MessageListener.h, 4164
 src/main/cms/MessageNotReadableException.h, 4164
 src/main/cms/MessageNotWriteableException.h, 4164
 src/main/cms/MessageProducer.h, 4165
 src/main/cms/Queue.h, 4166
 src/main/cms/QueueBrowser.h, 4167
 src/main/cms/Session.h, 4167
 src/main/cms/Startable.h, 4168
 src/main/cms/StreamMessage.h, 4169
 src/main/cms/TemporaryQueue.h, 4169
 src/main/cms/TemporaryTopic.h, 4170
 src/main/cms/TextMessage.h, 4170
 src/main/cms/Topic.h, 4171
 src/main/cms/UnsupportedOperationException.h, 4171
 src/main/decap/internal/AprPool.h, 4172
 src/main/decap/internal/DecafRuntime.h, 4173
 src/main/decap/internal/io/StandardErrorOutputStream.h, 4173
 src/main/decap/internal/io/StandardInputStream.h, 4174
 src/main/decap/internal/io/StandardOutputStream.h, 4174
 src/main/decap/internal/net/DefaultServerSocketFactory.h, 4174

src/main/decaf/internal/net/DefaultSocketFactory.h, 4175
 src/main/decaf/internal/net/Network.h, 4175
 src/main/decaf/internal/net/SocketFileDescriptor.h, 4176
 src/main/decaf/internal/net/ssl/DefaultSSLContext.h, 4176
 src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h, 4177
 src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h, 4177
 src/main/decaf/internal/net/ssl/openssl/OpenSSLContext.h, 4178
 src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h, 4179
 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h, 4179
 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h, 4180
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h, 4180
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h, 4181
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h, 4181
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h, 4182
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h, 4182
 src/main/decaf/internal/net/tcp/TcpSocket.h, 4183
 src/main/decaf/internal/net/tcp/TcpSocketInputStream.h, 4184
 src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h, 4184
 src/main/decaf/internal/net/URIEncoderDecoder.h, 4185
 src/main/decaf/internal/net/URIHelper.h, 4185
 src/main/decaf/internal/net/URIType.h, 4186
 src/main/decaf/internal/nio/BufferFactory.h, 4186
 src/main/decaf/internal/nio/ByteBuffer.h, 4187
 src/main/decaf/internal/nio/CharArrayBuffer.h, 4188
 src/main/decaf/internal/nio/DoubleArrayBuffer.h, 4188
 src/main/decaf/internal/nio/FloatArrayBuffer.h, 4189
 src/main/decaf/internal/nio/IntArrayBuffer.h, 4189
 src/main/decaf/internal/nio/LongArrayBuffer.h, 4190
 src/main/decaf/internal/nio/ShortArrayBuffer.h, 4191
 src/main/decaf/internal/security/unix/SecureRandomImpl.h, 4191
 src/main/decaf/internal/security/windows/SecureRandomImpl.h, 4192
 src/main/decaf/internal/util/ByteArrayAdapter.h, 4192
 src/main/decaf/internal/util/concurrent/ConditionImpl.h, 4193
 src/main/decaf/internal/util/concurrent/MutexImpl.h, 4193
 src/main/decaf/internal/util/concurrent/SynchronizableImpl.h, 4194
 src/main/decaf/internal/util/concurrent/Transferer.h, 4194
 src/main/decaf/internal/util/concurrent/TransferQueue.h, 4194
 src/main/decaf/internal/util/concurrent/TransferStack.h, 4195
 src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h, 4196
 src/main/decaf/internal/util/concurrent/unix/MutexHandle.h, 4197
 src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h, 4198
 src/main/decaf/internal/util/concurrent/windows/MutexHandle.h, 4198
 src/main/decaf/internal/util/Resource.h, 4198
 src/main/decaf/internal/util/ResourceLifecycleManager.h, 3805
 src/main/decaf/internal/util/TimerTaskHeap.h, 4199
 src/main/decaf/internal/util/zip/crc32.h, 4199
 src/main/decaf/internal/util/zip/deflate.h, 4199
 src/main/decaf/internal/util/zip/gzguts.h, 4202
 src/main/decaf/internal/util/zip/inffast.h, 4204
 src/main/decaf/internal/util/zip/inffixed.h, 4205
 src/main/decaf/internal/util/zip/inflate.h, 4205
 src/main/decaf/internal/util/zip/inftrees.h, 4206
 src/main/decaf/internal/util/zip/trees.h, 4207
 src/main/decaf/internal/util/zip/zconf.h, 4209
 src/main/decaf/internal/util/zip/zlib.h, 4211
 src/main/decaf/internal/util/zip/zutil.h, 4215

- src/main/decaf/io/BlockingByteArrayInputStream.h, 4217
- src/main/decaf/io/BufferedInputStream.h, 4218
- src/main/decaf/io/BufferedOutputStream.h, 4218
- src/main/decaf/io/ByteArrayInputStream.h, 4219
- src/main/decaf/io/ByteArrayOutputStream.h, 4219
- src/main/decaf/io/Closeable.h, 4155
- src/main/decaf/io/DataInput.h, 4220
- src/main/decaf/io/DataInputStream.h, 4220
- src/main/decaf/io/DataOutput.h, 4221
- src/main/decaf/io/DataOutputStream.h, 4222
- src/main/decaf/io/EOFException.h, 4222
- src/main/decaf/io/FileDescriptor.h, 4222
- src/main/decaf/io/FilterInputStream.h, 4223
- src/main/decaf/io/FilterOutputStream.h, 4223
- src/main/decaf/io/Flushable.h, 4224
- src/main/decaf/io/InputStream.h, 4224
- src/main/decaf/io/InputStreamReader.h, 4225
- src/main/decaf/io/InterruptedIOException.h, 4225
- src/main/decaf/io/IOException.h, 4226
- src/main/decaf/io/OutputStream.h, 4226
- src/main/decaf/io/OutputStreamWriter.h, 4227
- src/main/decaf/io/PushbackInputStream.h, 4227
- src/main/decaf/io/Reader.h, 4228
- src/main/decaf/io/UnsupportedEncodingException.h, 4228
- src/main/decaf/io/UTFDataFormatException.h, 4229
- src/main/decaf/io/Writer.h, 4229
- src/main/decaf/lang/Appendable.h, 4230
- src/main/decaf/lang/ArrayPointer.h, 4230
- src/main/decaf/lang/Boolean.h, 4231
- src/main/decaf/lang/Byte.h, 4232
- src/main/decaf/lang/Character.h, 4232
- src/main/decaf/lang/CharSequence.h, 4232
- src/main/decaf/lang/Comparable.h, 4233
- src/main/decaf/lang/Double.h, 4233
- src/main/decaf/lang/Exception.h, 4234
- src/main/decaf/lang/exceptions/ClassCastException.h, 4234
- src/main/decaf/lang/exceptions/ExceptionDefinition.h, 3856
- src/main/decaf/lang/exceptions/IllegalArgumentException.h, 4235
- src/main/decaf/lang/exceptions/IllegalMonitorStateException.h, 4235
- src/main/decaf/lang/exceptions/IllegalStateException.h, 4160
- src/main/decaf/lang/exceptions/IllegalThreadStateException.h, 4235
- src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h, 4236
- src/main/decaf/lang/exceptions/InterruptedException.h, 4236
- src/main/decaf/lang/exceptions/InvalidStateException.h, 4237
- src/main/decaf/lang/exceptions/NoSuchElementException.h, 4237
- src/main/decaf/lang/exceptions/NullPointerException.h, 4237
- src/main/decaf/lang/exceptions/NumberFormatException.h, 4238
- src/main/decaf/lang/exceptions/RuntimeException.h, 4238
- src/main/decaf/lang/exceptions/UnsupportedOperationException.h, 4172
- src/main/decaf/lang/Float.h, 4239
- src/main/decaf/lang/Integer.h, 4239
- src/main/decaf/lang/Iterable.h, 4239
- src/main/decaf/lang/Long.h, 4240
- src/main/decaf/lang/Math.h, 4240
- src/main/decaf/lang/Number.h, 4241
- src/main/decaf/lang/Pointer.h, 4241
- src/main/decaf/lang/Readable.h, 4242
- src/main/decaf/lang/Runnable.h, 4243
- src/main/decaf/lang/Runtime.h, 4243
- src/main/decaf/lang/Short.h, 4244
- src/main/decaf/lang/String.h, 4244
- src/main/decaf/lang/System.h, 4244
- src/main/decaf/lang/Thread.h, 4245
- src/main/decaf/lang/ThreadGroup.h, 4246
- src/main/decaf/lang/Throwable.h, 4246
- src/main/decaf/net/BindException.h, 4247
- src/main/decaf/net/ConnectException.h, 4247
- src/main/decaf/net/HttpRetryException.h, 4247
- src/main/decaf/net/Inet4Address.h, 4248
- src/main/decaf/net/Inet6Address.h, 4248
- src/main/decaf/net/InetAddress.h, 4249
- src/main/decaf/net/InetSocketAddress.h, 4249
- src/main/decaf/net/MalformedURLException.h, 4249
- src/main/decaf/net/NoRouteToHostException.h, 4250
- src/main/decaf/net/PortUnreachableException.h, 4250
- src/main/decaf/net/ProtocolException.h, 4251
- src/main/decaf/net/ServerSocket.h, 4251
- src/main/decaf/net/ServerSocketFactory.h, 4252

- src/main/decaf/net/Socket.h, 4252
- src/main/decaf/net/SocketAddress.h, 4253
- src/main/decaf/net/SocketError.h, 4253
- src/main/decaf/net/SocketException.h, 4254
- src/main/decaf/net/SocketFactory.h, 4254
- src/main/decaf/net/SocketImpl.h, 4255
- src/main/decaf/net/SocketImplFactory.h, 4255
- src/main/decaf/net/SocketOptions.h, 4256
- src/main/decaf/net/SocketTimeoutException.h, 4256
- src/main/decaf/net/ssl/SSLContext.h, 4256
- src/main/decaf/net/ssl/SSLContextSpi.h, 4257
- src/main/decaf/net/ssl/SSLParameters.h, 4257
- src/main/decaf/net/ssl/SSLServerSocket.h, 4258
- src/main/decaf/net/ssl/SSLServerSocketFactory.h, 4258
- src/main/decaf/net/ssl/SSLSocket.h, 4259
- src/main/decaf/net/ssl/SSLSocketFactory.h, 4259
- src/main/decaf/net/UnknownHostException.h, 4260
- src/main/decaf/net/UnknownServiceException.h, 4260
- src/main/decaf/net/URI.h, 4261
- src/main/decaf/net/URISyntaxException.h, 4261
- src/main/decaf/net/URL.h, 4262
- src/main/decaf/net/URLDecoder.h, 4262
- src/main/decaf/net/URLEncoder.h, 4262
- src/main/decaf/nio/Buffer.h, 4263
- src/main/decaf/nio/BufferOverflowException.h, 4263
- src/main/decaf/nio/BufferUnderflowException.h, 4264
- src/main/decaf/nio/ByteBuffer.h, 4264
- src/main/decaf/nio/CharBuffer.h, 4265
- src/main/decaf/nio/DoubleBuffer.h, 4265
- src/main/decaf/nio/FloatBuffer.h, 4266
- src/main/decaf/nio/IntBuffer.h, 4266
- src/main/decaf/nio/InvalidMarkException.h, 4267
- src/main/decaf/nio/LongBuffer.h, 4267
- src/main/decaf/nio/ReadOnlyBufferException.h, 4268
- src/main/decaf/nio/ShortBuffer.h, 4268
- src/main/decaf/security/auth/x500/X500Principal.h, 4269
- src/main/decaf/security/cert/Certificate.h, 4269
- src/main/decaf/security/cert/CertificateEncodingException.h, 4270
- src/main/decaf/security/cert/CertificateException.h, 4270
- src/main/decaf/security/cert/CertificateExpiredException.h, 4271
- src/main/decaf/security/cert/CertificateNotYetValidException.h, 4271
- src/main/decaf/security/cert/CertificateParsingException.h, 4272
- src/main/decaf/security/cert/X509Certificate.h, 4272
- src/main/decaf/security/GeneralSecurityException.h, 4273
- src/main/decaf/security/InvalidKeyException.h, 4273
- src/main/decaf/security/Key.h, 4273
- src/main/decaf/security/KeyException.h, 4274
- src/main/decaf/security/KeyManagementException.h, 4274
- src/main/decaf/security/NoSuchAlgorithmException.h, 4275
- src/main/decaf/security/NoSuchProviderException.h, 4275
- src/main/decaf/security/Principal.h, 4275
- src/main/decaf/security/PublicKey.h, 4276
- src/main/decaf/security/SecureRandom.h, 4276
- src/main/decaf/security/SecureRandomSpi.h, 4277
- src/main/decaf/security/SignatureException.h, 4277
- src/main/decaf/util/AbstractCollection.h, 4278
- src/main/decaf/util/AbstractList.h, 4278
- src/main/decaf/util/AbstractMap.h, 4279
- src/main/decaf/util/AbstractQueue.h, 4279
- src/main/decaf/util/AbstractSequentialList.h, 4280
- src/main/decaf/util/AbstractSet.h, 4281
- src/main/decaf/util/Collection.h, 4281
- src/main/decaf/util/Comparator.h, 4282
- src/main/decaf/util/comparators/Less.h, 4282
- src/main/decaf/util/concurrent/atomic/AtomicBoolean.h, 4283
- src/main/decaf/util/concurrent/atomic/AtomicInteger.h, 4283
- src/main/decaf/util/concurrent/atomic/AtomicReferenceCounter.h, 4284
- src/main/decaf/util/concurrent/atomic/AtomicReference.h, 4284
- src/main/decaf/util/concurrent/BlockingQueue.h, 4285
- src/main/decaf/util/concurrent/BrokenBarrierException.h, 4285
- src/main/decaf/util/concurrent/Callable.h, 4286
- src/main/decaf/util/concurrent/CancellationException.h, 4286

src/main/decaf/util/concurrent/Concurrent.h, 4287
 src/main/decaf/util/concurrent/ConcurrentMap.h, 4288
 src/main/decaf/util/concurrent/ConcurrentStlMap.h, 4288
 src/main/decaf/util/concurrent/CountDownLatch.h, 4289
 src/main/decaf/util/concurrent/Delayed.h, 4289
 src/main/decaf/util/concurrent/ExecutionException.h, 4290
 src/main/decaf/util/concurrent/Executor.h, 4290
 src/main/decaf/util/concurrent/ExecutorService.h, 4291
 src/main/decaf/util/concurrent/Future.h, 4291
 src/main/decaf/util/concurrent/Lock.h, 4292
 src/main/decaf/util/concurrent/locks/Condition.h, 4293
 src/main/decaf/util/concurrent/locks/Lock.h, 4292
 src/main/decaf/util/concurrent/locks/LockSupport.h, 4293
 src/main/decaf/util/concurrent/locks/ReadWriteLock.h, 4294
 src/main/decaf/util/concurrent/locks/ReentrantLock.h, 4294
 src/main/decaf/util/concurrent/Mutex.h, 4295
 src/main/decaf/util/concurrent/PooledThread.h, 4295
 src/main/decaf/util/concurrent/PooledThreadListener.h, 4296
 src/main/decaf/util/concurrent/RejectedExecutionException.h, 4296
 src/main/decaf/util/concurrent/RejectedExecutionHandler.h, 4296
 src/main/decaf/util/concurrent/Semaphore.h, 4297
 src/main/decaf/util/concurrent/Synchronizable.h, 4297
 src/main/decaf/util/concurrent/SynchronousQueue.h, 4298
 src/main/decaf/util/concurrent/TaskListener.h, 4299
 src/main/decaf/util/concurrent/ThreadFactory.h, 4299
 src/main/decaf/util/concurrent/ThreadPool.h, 4299
 src/main/decaf/util/concurrent/TimeoutException.h, 4300
 src/main/decaf/util/concurrent/TimeUnit.h, 4301
 src/main/decaf/util/Config.h, 3889
 src/main/decaf/util/Date.h, 4301
 src/main/decaf/util/Iterator.h, 4302
 src/main/decaf/util/List.h, 4302
 src/main/decaf/util/ListIterator.h, 4303
 src/main/decaf/util/logging/ConsoleHandler.h, 4303
 src/main/decaf/util/logging/ErrorHandler.h, 4304
 src/main/decaf/util/logging/Filter.h, 4304
 src/main/decaf/util/logging/Formatter.h, 4305
 src/main/decaf/util/logging/Handler.h, 4305
 src/main/decaf/util/logging/Level.h, 4306
 src/main/decaf/util/logging/Logger.h, 4306
 src/main/decaf/util/logging/LoggerCommon.h, 4307
 src/main/decaf/util/logging/LoggerDefines.h, 4307
 src/main/decaf/util/logging/LoggerHierarchy.h, 4309
 src/main/decaf/util/logging/LogManager.h, 4309
 src/main/decaf/util/logging/LogRecord.h, 4310
 src/main/decaf/util/logging/LogWriter.h, 4311
 src/main/decaf/util/logging/MarkBlockLogger.h, 4311
 src/main/decaf/util/logging/PropertiesChangeListener.h, 4312
 src/main/decaf/util/logging/SimpleFormatter.h, 4312
 src/main/decaf/util/logging/SimpleLogger.h, 4313
 src/main/decaf/util/logging/StreamHandler.h, 4313
 src/main/decaf/util/logging/XMLFormatter.h, 4314
 src/main/decaf/util/Map.h, 4314
 src/main/decaf/util/PriorityQueue.h, 4315
 src/main/decaf/util/Properties.h, 4315
 src/main/decaf/util/Queue.h, 4166
 src/main/decaf/util/Random.h, 4316
 src/main/decaf/util/Set.h, 4316
 src/main/decaf/util/StlList.h, 4317
 src/main/decaf/util/StlMap.h, 4318
 src/main/decaf/util/StlQueue.h, 4318
 src/main/decaf/util/StlSet.h, 4319
 src/main/decaf/util/StringTokenizer.h, 4319
 src/main/decaf/util/Timer.h, 4320
 src/main/decaf/util/TimerTask.h, 4320
 src/main/decaf/util/UUID.h, 4321
 src/main/decaf/util/zip/Adler32.h, 4321
 src/main/decaf/util/zip/CheckedInputStream.h, 4322

- src/main/decaf/util/zip/CheckedOutputStream.h, 4322
- src/main/decaf/util/zip/Checksum.h, 4323
- src/main/decaf/util/zip/CRC32.h, 4323
- src/main/decaf/util/zip/DataFormatException.h, 4324
- src/main/decaf/util/zip/Deflater.h, 4324
- src/main/decaf/util/zip/DeflaterOutputStream.h, 4325
- src/main/decaf/util/zip/Inflater.h, 4325
- src/main/decaf/util/zip/InflaterInputStream.h, 4326
- src/main/decaf/util/zip/ZipException.h, 4326
- SSLContext
 - decaf::net::ssl::SSLContext, 3322
- SSLParameters
 - decaf::net::ssl::SSLParameters, 3327
- SSLServerSocket
 - decaf::net::ssl::SSLServerSocket, 3331
- SSLServerSocketFactory
 - decaf::net::ssl::SSLServerSocketFactory, 3336
- SSLSocket
 - decaf::net::ssl::SSLSocket, 3339, 3340
- SSLSocketFactory
 - decaf::net::ssl::SSLSocketFactory, 3346
- SslTransport
 - activemq::transport::tcp::SslTransport, 3348
- stackTrace
 - decaf::lang::Exception, 1718
- StandardErrorOutputStream
 - decaf::internal::io::StandardErrorOutputStream, 3352
- StandardInputStream
 - decaf::internal::io::StandardInputStream, 3353
- StandardOutputStream
 - decaf::internal::io::StandardOutputStream, 3354
- start
 - activemq::commands::ConsumerControl, 1306
 - activemq::core::ActiveMQConnection, 250
 - activemq::core::ActiveMQConsumer, 277
 - activemq::core::ActiveMQSession, 481
 - activemq::core::ActiveMQSessionExecutor, 485
 - activemq::core::MessageDispatchChannel, 2436
 - activemq::transport::correlator::ResponseCorrelator, 3084
 - activemq::transport::failover::FailoverTransport, 1763
 - activemq::transport::IOTransport, 2011
 - activemq::transport::mock::MockTransport, 2601
 - activemq::transport::Transport, 3633
 - activemq::transport::TransportFilter, 3642
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2716
 - cms::Startable, 3356
 - decaf::lang::Thread, 3528
 - gz_state, 1851
 - startHandshake
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2685
 - decaf::net::ssl::SSLSocket, 3344
- stat_desc
 - tree_desc_s, 3648
- State
 - decaf::lang::Thread, 3523
- state
 - z_stream_s, 3795
- static_dtrees
 - trees.h, 4209
- static_len
 - internal_state, 1985
- static_ltree
 - trees.h, 4209
- static_tree_desc
 - deflate.h, 4202
- STATIC_TREES
 - zutil.h, 4217
- staticCast
 - decaf::lang::Pointer, 2763
- StaticInitializer
 - activemq::core::ActiveMQConstants::StaticInitializer, 3357
- status
 - internal_state, 1985
- std, 141
- std::binary_function, 768
- std::less< decaf::lang::ArrayPointer< T > >, 2184
 - operator(), 2184
- std::less< decaf::lang::Pointer< T > >, 2185
 - operator(), 2185
- StlList
 - decaf::util::StlList, 3362
- StlMap
 - decaf::util::StlMap, 3373
- StlQueue
 - decaf::util::StlQueue, 3384
- StlSet
 - decaf::util::StlSet, 3392
- StompFrame

- activemq::wireformat::stomp::StompFrame, 3399
- StompHelper
 - activemq::wireformat::stomp::StompHelper, 3404
- StompWireFormat
 - activemq::wireformat::stomp::StompWireFormat, 3408
- StompWireFormatFactory
 - activemq::wireformat::stomp::StompWireFormatFactory, 3411
- stop
 - activemq::commands::ConsumerControl, 1306
 - activemq::core::ActiveMQConnection, 250
 - activemq::core::ActiveMQConsumer, 277
 - activemq::core::ActiveMQSession, 481
 - activemq::core::ActiveMQSessionExecutor, 485
 - activemq::core::MessageDispatchChannel, 2436
 - activemq::transport::failover::FailoverTransport, 1763
 - activemq::transport::IOTransport, 2011
 - activemq::transport::mock::MockTransport, 2601
 - activemq::transport::Transport, 3634
 - activemq::transport::TransportFilter, 3643
 - cms::Stoppable, 3412
 - decaf::util::concurrent::PooledThread, 2779
- store
 - decaf::util::Properties, 2934
- STORED
 - inflate.h, 4206
- STORED_BLOCK
 - zutil.h, 4217
- strategy
 - gz_state, 1851
 - internal_state, 1985
- StreamHandler
 - decaf::util::logging::StreamHandler, 3413
- String
 - decaf::lang::String, 3429
- STRING_TYPE
 - activemq::util::PrimitiveValueNode, 2821
- StringTokenizer
 - decaf::util::StringTokenizer, 3431
- stringValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2815
- strm
 - gz_state, 1851
 - internal_state, 1985
- strstart
 - internal_state, 1985
- subscriptionName
 - activemq::commands::RemoveSubscriptionInfo, 3019
 - activemq::commands::SubscriptionInfo, 3438
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- subscribedDestination
 - activemq::commands::SubscriptionInfo, 3438
- SubscriptionInfo
 - activemq::commands::SubscriptionInfo, 3435
- SubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::SubscriptionInfo, 3443
 - activemq::wireformat::openwire::marshal::v2::SubscriptionInfo, 3459
 - activemq::wireformat::openwire::marshal::v3::SubscriptionInfo, 3439
 - activemq::wireformat::openwire::marshal::v4::SubscriptionInfo, 3451
 - activemq::wireformat::openwire::marshal::v5::SubscriptionInfo, 3447
 - activemq::wireformat::openwire::marshal::v6::SubscriptionInfo, 3455
- subscriptionName
 - activemq::commands::ConsumerInfo, 1365
- subscriptionName
 - activemq::commands::JournalTopicAck, 2045
- subSequence
 - decaf::internal::nio::CharArrayBuffer, 1036
 - decaf::lang::CharSequence, 1054
 - decaf::lang::String, 3429
 - decaf::nio::CharBuffer, 1051
- supportsUrgentData
 - decaf::net::SocketImpl, 3313
- suspend
 - activemq::commands::ConnectionControl, 1176
- swap
 - decaf::lang::ArrayPointer, 675
 - decaf::lang::Pointer, 2763
 - decaf::util::concurrent::atomic::AtomicRefCounter, 687
- SYNC
 - inflate.h, 4206
- sync
 - decaf::io::FileDescriptor, 1769
- SynchronizableImpl

- decaf::internal::util::concurrent::SynchronizableImp
 - 3474
- synchronized
 - Concurrent.h, 4287
- SynchronousQueue
 - decaf::util::concurrent::SynchronousQueue, 3480
- syncRequest
 - activemq::core::ActiveMQConnection, 250
 - activemq::core::ActiveMQSession, 481
- System
 - decaf::lang::System, 3489
- TABLE
 - inflate.h, 4206
- take
 - decaf::util::concurrent::BlockingQueue, 780
 - decaf::util::concurrent::SynchronousQueue, 3486
- takeSession
 - activemq::cmsutil::SessionPool, 3217
- targetConsumerId
 - activemq::commands::Message, 2374
- Task
 - decaf::util::concurrent::ThreadPool, 3533
- TcpSocket
 - decaf::internal::net::tcp::TcpSocket, 3500
- TcpSocketInputStream
 - decaf::internal::net::tcp::TcpSocketInputStream, 3507
- TcpSocketOutputStream
 - decaf::internal::net::tcp::TcpSocketOutputStream, 3509
- TcpTransport
 - activemq::transport::tcp::TcpTransport, 3511
- TEMP_POSTFIX
 - activemq::commands::ActiveMQDestination, 289
- TEMP_PREFIX
 - activemq::commands::ActiveMQDestination, 289
- TEMP_QUEUE_QUALIFED_PREFIX
 - activemq::commands::ActiveMQDestination, 289
- TEMP_TOPIC_QUALIFED_PREFIX
 - activemq::commands::ActiveMQDestination, 289
- TEMPORARY_QUEUE
 - cms::Destination, 1611
- TEMPORARY_TOPIC
 - cms::Destination, 1611
- TEMPQUEUE_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- TEMPTOPIC_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- TERMINATED
 - decaf::lang::Thread, 3524
- TEXT
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- text
 - activemq::commands::ActiveMQTextMessage, 609
 - gz_header_s, 1849
- Thread
 - decaf::lang::Thread, 3524
- ThreadGroup
 - decaf::lang::ThreadGroup, 3531
- ThreadPool
 - decaf::util::concurrent::ThreadPool, 3533
- Throwable
 - decaf::lang::Throwable, 3538
- Throwing
 - decaf::util::logging, 140
- throwing
 - decaf::util::logging::Logger, 2248
- tightMarshal
 - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 753
 - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1532
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMarshaller, 177
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMarshaller, 216
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 296
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMarshaller, 335
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 361
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMarshaller, 405
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 447
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMarshaller, 506
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 532
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 559
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 591

activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	2417
619	
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMessageMarshaller	2456
647	
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	2485
718	
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	2521
811	
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	2542
842	
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	2586
1187	
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2639
1218	
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	2754
1248	
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	2865
1278	
activemq::wireformat::openwire::marshal::v1::ConsumerGroupMarshaller	2896
1321	
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	2912
1348	
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	3005
1380	
activemq::wireformat::openwire::marshal::v1::ContactCommandMarshaller	3021
1407	
activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller	3052
1440	
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	3105
1502	
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	3187
1632	
activemq::wireformat::openwire::marshal::v1::DiscardResponseMarshaller	3202
1665	
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	3262
1745	
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3444
1833	
activemq::wireformat::openwire::marshal::v1::IntegrationResponseMarshaller	3578
1977	
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3605
2039	
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	3746
2068	
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	3783
2090	
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaller	185
2121	
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	232
2147	
activemq::wireformat::openwire::marshal::v1::LastReceivedCommandMarshaller	308
2180	
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	347
2226	

activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	2023	activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	2052
373		activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	2052
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2074	activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	2052
417		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2074
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2105	activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller	2074
459		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2105
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2131	activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	2105
518		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2131
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2168	activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	2131
543		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2168
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2210	activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	2168
571		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2210
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2405	activemq::wireformat::openwire::marshal::v2::LocalTransactionMarshaller	2210
599		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2405
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2440	activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	2405
631		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2440
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2473	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2440
659		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2473
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	2501	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2473
738		activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	2501
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2534	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2501
823		activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2534
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2570	activemq::wireformat::openwire::marshal::v2::MessageMarshaller	2534
854		activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2570
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2619	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller	2570
1199		activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2619
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2738	activemq::wireformat::openwire::marshal::v2::NetworkBridgeMarshaller	2619
1206		activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2738
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	2845	activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	2738
1236		activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	2845
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	2876	activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	2845
1266		activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	2876
activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	2908	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	2876
1309		activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	2908
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	2993	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	2908
1336		activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	2993
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	3029	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	2993
1368		activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	3029
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	3056	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller	3029
1395		activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	3056
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	3092	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	3056
1428		activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	3092
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	3167	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	3092
1490		activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	3167
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	3210	activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	3167
1620		activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	3210
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	3258	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	3210
1653		activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	3258
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	3460	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	3258
1729		activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	3460
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	3582	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	3460
1821		activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	3582
activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller		activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller	3582
1965		activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller	

activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	1432
3621	
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	1494
3738	
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	1624
3775	
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobWireFormatMessageMarshaller	1657
173	
activemq::wireformat::openwire::marshal::v3::ActiveMQByteWireFormatMessageMarshaller	1733
212	
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationFormatMarshaller	1825
292	
activemq::wireformat::openwire::marshal::v3::ActiveMQMapWireFormatMessageMarshaller	1969
331	
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	2031
357	
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectWireFormatMessageMarshaller	2056
401	
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueWireFormatMarshaller	2078
443	
activemq::wireformat::openwire::marshal::v3::ActiveMQStackWireFormatMessageMarshaller	2109
502	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueFormatMarshaller	2135
529	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueFormatMarshaller	2164
555	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	2214
583	
activemq::wireformat::openwire::marshal::v3::ActiveMQTextWireFormatMessageMarshaller	2409
611	
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	2444
639	
activemq::wireformat::openwire::marshal::v3::BaseCommandWireFormatMarshaller	2477
704	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2513
803	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2530
834	
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2578
1179	
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2631
1210	
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2746
1240	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	2853
1270	
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller	2884
1313	
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	2920
1340	
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	3001
1372	
activemq::wireformat::openwire::marshal::v3::ContentCommandMarshaller	3025
1399	

activemq::wireformat::openwire::marshal::v3::ReplyCommandMarshaller	openwire::marshal::v4::ConnectionError
3060	1214
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	openwire::marshal::v4::ConnectionIdMar
3101	1244
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	openwire::marshal::v4::ConnectionInfoM
3183	1274
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	openwire::marshal::v4::ConsumerContro
3206	1317
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	openwire::marshal::v4::ConsumerIdMar
3270	1344
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	openwire::marshal::v4::ConsumerInfoM
3441	1376
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	openwire::marshal::v4::ControlComm
3586	1403
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	openwire::marshal::v4::DataArrayRespo
3609	1436
activemq::wireformat::openwire::marshal::v3::WireFormatMarshaller	openwire::marshal::v4::DataResponseM
3750	1498
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	openwire::marshal::v4::DestinationInfoM
3787	1628
activemq::wireformat::openwire::marshal::v4::ActiveMQBlockWireFormatMarshaller	openwire::marshal::v4::DiscoveryEventM
181	1661
activemq::wireformat::openwire::marshal::v4::ActiveMQByteWireFormatMarshaller	openwire::marshal::v4::ExceptionRespo
220	1741
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	openwire::marshal::v4::FlushCommandL
300	1829
activemq::wireformat::openwire::marshal::v4::ActiveMQMapWireFormatMarshaller	openwire::marshal::v4::IntegerResponse
339	1973
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	openwire::marshal::v4::JournalQueueAc
365	2035
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectWireFormatMarshaller	openwire::marshal::v4::JournalTopicAck
409	2064
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	openwire::marshal::v4::JournalTraceMa
451	2086
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamingMessageMarshaller	openwire::marshal::v4::JournalTransact
510	2117
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	openwire::marshal::v4::KeepAliveInfoM
536	2139
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	openwire::marshal::v4::LastPartialCom
563	2176
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	openwire::marshal::v4::LocalTransaction
587	2222
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	openwire::marshal::v4::MessageAckMar
615	2413
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	openwire::marshal::v4::MessageDispatc
643	2452
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	openwire::marshal::v4::MessageDispatc
711	2481
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	openwire::marshal::v4::MessageIdMarsh
807	2505
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	openwire::marshal::v4::MessageMarshal
838	2538
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	openwire::marshal::v4::MessagePullMar
1183	2582

activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller	openwire::marshal::v5::ActiveMQTemp
2635	595
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	openwire::marshal::v5::ActiveMQTextM
2750	623
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	openwire::marshal::v5::ActiveMQTopicC
2849	651
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	openwire::marshal::v5::BaseCommandM
2880	724
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	openwire::marshal::v5::BrokerIdMarsh
2904	815
activemq::wireformat::openwire::marshal::v4::RemoteInfoMarshaller	openwire::marshal::v5::BrokerInfoMarsh
3013	846
activemq::wireformat::openwire::marshal::v4::RemoteSubscriptionInfoMarshaller	openwire::marshal::v5::ConnectionCont
3041	1191
activemq::wireformat::openwire::marshal::v4::ReplaceCommandMarshaller	openwire::marshal::v5::ConnectionError
3048	1222
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	openwire::marshal::v5::ConnectionIdMa
3087	1252
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	openwire::marshal::v5::ConnectionInfoM
3171	1282
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	openwire::marshal::v5::ConsumerContro
3214	1325
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	openwire::marshal::v5::ConsumerIdMar
3274	1352
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	openwire::marshal::v5::ConsumerInfoMa
3452	1384
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	openwire::marshal::v5::ControlComman
3589	1411
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	openwire::marshal::v5::DataArrayRespo
3617	1444
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	openwire::marshal::v5::DataResponseM
3742	1482
activemq::wireformat::openwire::marshal::v4::XATransactionMarshaller	openwire::marshal::v5::DestinationInfoM
3779	1640
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobWireFormatMarshaller	openwire::marshal::v5::DiscoveryEventM
189	1669
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	openwire::marshal::v5::ExceptionRespo
224	1737
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	openwire::marshal::v5::FlushCommandL
304	1837
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	openwire::marshal::v5::IntegerResponse
343	1981
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	openwire::marshal::v5::JournalQueueAc
369	2027
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	openwire::marshal::v5::JournalTopicAck
413	2048
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	openwire::marshal::v5::JournalTraceMa
455	2094
activemq::wireformat::openwire::marshal::v5::ActiveMQSequenceMessageMarshaller	openwire::marshal::v5::JournalTransact
514	2113
activemq::wireformat::openwire::marshal::v5::ActiveMQTempRedundantMarshaller	openwire::marshal::v5::KeepAliveInfoM
540	2143
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	openwire::marshal::v5::LastPartialComr
567	2172

activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller	2218	activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessage	351
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller	2421	activemq::wireformat::openwire::marshal::v6::ActiveMQMessage	377
activemq::wireformat::openwire::marshal::v5::MessageDispatchFlusher	2448	activemq::wireformat::openwire::marshal::v6::ActiveMQObject	421
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	2489	activemq::wireformat::openwire::marshal::v6::ActiveMQQueue	463
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller	2509	activemq::wireformat::openwire::marshal::v6::ActiveMQStream	522
activemq::wireformat::openwire::marshal::v5::MessageMarshaller	2525	activemq::wireformat::openwire::marshal::v6::ActiveMQTemplate	547
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller	2574	activemq::wireformat::openwire::marshal::v6::ActiveMQTemp	575
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller	2627	activemq::wireformat::openwire::marshal::v6::ActiveMQTemp	603
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	2742	activemq::wireformat::openwire::marshal::v6::ActiveMQTextM	627
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller	2857	activemq::wireformat::openwire::marshal::v6::ActiveMQTopic	655
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	2888	activemq::wireformat::openwire::marshal::v6::BaseCommandM	731
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	2916	activemq::wireformat::openwire::marshal::v6::BrokerIdMarsh	819
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller	3009	activemq::wireformat::openwire::marshal::v6::BrokerInfoMarsh	850
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	3037	activemq::wireformat::openwire::marshal::v6::ConnectionCont	1195
activemq::wireformat::openwire::marshal::v5::ReplyCommandMarshaller	3068	activemq::wireformat::openwire::marshal::v6::ConnectionError	1226
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	3096	activemq::wireformat::openwire::marshal::v6::ConnectionIdMa	1256
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	3179	activemq::wireformat::openwire::marshal::v6::ConnectionInfoM	1286
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	3198	activemq::wireformat::openwire::marshal::v6::ConsumerContro	1329
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	3266	activemq::wireformat::openwire::marshal::v6::ConsumerIdMar	1356
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	3448	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMa	1388
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller	3575	activemq::wireformat::openwire::marshal::v6::ControlComman	1415
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	3601	activemq::wireformat::openwire::marshal::v6::DataArrayRespo	1448
activemq::wireformat::openwire::marshal::v5::WireFormatWireMarshaller	3730	activemq::wireformat::openwire::marshal::v6::DataResponseM	1486
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	3791	activemq::wireformat::openwire::marshal::v6::DestinationInfoM	1636
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobWireFormatMarshaller	193	activemq::wireformat::openwire::marshal::v6::DiscoveryEventM	1649
activemq::wireformat::openwire::marshal::v6::ActiveMQByteMessageMarshaller	228	activemq::wireformat::openwire::marshal::v6::ExceptionRespor	1725
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller	312	activemq::wireformat::openwire::marshal::v6::FlushCommandD	1817

activemq::wireformat::openwire::marshal::v6::IntegerResponseInfoMarshaller, 1961
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, 2019
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, 2060
 activemq::wireformat::openwire::marshal::v6::JournalTransInfoMarshaller, 2082
 activemq::wireformat::openwire::marshal::v6::LightMessageActionMarshaller, 2101
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2127
 activemq::wireformat::openwire::marshal::v6::LastPartCommandMarshaller, 2161
 activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2206
 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2401
 activemq::wireformat::openwire::marshal::v6::MessagePatchMarshaller, 2460
 activemq::wireformat::openwire::marshal::v6::MessagePatchNotificationMarshaller, 2469
 activemq::wireformat::openwire::marshal::v6::MessageRefMarshaller, 2517
 activemq::wireformat::openwire::marshal::v6::MessageRefMarshaller, 2546
 activemq::wireformat::openwire::marshal::v6::MessageRefMarshaller, 2590
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2623
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2733
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 2861
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 2892
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 2924
 activemq::wireformat::openwire::marshal::v6::RemoveIdMarshaller, 2997
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3033
 activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3064
 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3110
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3175
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3194
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3254
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3456

1321	2896
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller
1348	2913
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller
1380	3006
activemq::wireformat::openwire::marshal::v1::ConsumerQueueIdMarshaller	activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller
1408	3022
activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller	activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
1441	3053
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v1::ResponseMarshaller
1502	3106
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller
1633	3187
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller	activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller
1665	3202
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller
1745	3263
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller
1833	3445
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller
1977	3579
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller
2039	3605
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller
2068	3746
activemq::wireformat::openwire::marshal::v1::JournalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller
2090	3784
activemq::wireformat::openwire::marshal::v1::JournalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller
2121	185
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller
2148	232
activemq::wireformat::openwire::marshal::v1::LastReceivedCommandMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
2181	308
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller
2227	347
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
2418	374
activemq::wireformat::openwire::marshal::v1::MessageDispatchInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller
2457	418
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
2485	460
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller
2521	518
activemq::wireformat::openwire::marshal::v1::MessageMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTemplateMarshaller
2542	544
activemq::wireformat::openwire::marshal::v1::MessagePushMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller
2587	571
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller
2639	599
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTextMarshaller
2755	632
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
2865	659
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller

739	2473
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller
823	2502
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	activemq::wireformat::openwire::marshal::v2::MessageMarshaller
854	2534
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller
1199	2571
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilter
1207	2619
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller
1237	2738
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller
1266	2845
activemq::wireformat::openwire::marshal::v2::ConsumerGroupViewMarshaller	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller
1309	2877
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller
1337	2909
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller
1368	2994
activemq::wireformat::openwire::marshal::v2::ContactCommandMarshaller	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller
1396	3030
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
1429	3057
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller
1490	3092
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller
1621	3168
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller
1653	3210
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller
1729	3259
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller
1821	3460
activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller	activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller
1965	3582
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller
2024	3621
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller
2052	3738
activemq::wireformat::openwire::marshal::v2::JournalTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::XATransactionInfoMarshaller
2075	3776
activemq::wireformat::openwire::marshal::v2::JournalTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller
2105	173
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMarshaller
2132	212
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
2169	292
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMarshaller
2211	331
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
2406	358
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMarshaller
2441	402
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller

444	2078
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueWireFormatMessageMarshaller;marshal::v3::JournalTransact	
502	2109
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueWireFormatMessageMarshaller;marshal::v3::KeepAliveInfoM	
529	2136
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueWireFormatMessageMarshaller;marshal::v3::LastPartialComm	
555	2165
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueWireFormatMessageMarshaller;marshal::v3::LocalTransaction	
583	2215
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueWireFormatMessageMarshaller;marshal::v3::MessageAckMar	
612	2410
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueWireFormatMessageMarshaller;openwire::marshal::v3::MessageDispat ch	
639	2445
activemq::wireformat::openwire::marshal::v3::BaseCommandWireFormat;openwire::marshal::v3::MessageDispat ch	
705	2477
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller;openwire::marshal::v3::MessageIdMarsh	
804	2513
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller;openwire::marshal::v3::MessageMarsh	
834	2530
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller;openwire::marshal::v3::MessagePullMar	
1179	2579
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller;openwire::marshal::v3::NetworkBridgeF	
1211	2631
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller;openwire::marshal::v3::PartialCommand	
1240	2747
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller;openwire::marshal::v3::ProducerAckMar	
1270	2853
activemq::wireformat::openwire::marshal::v3::ConsumerGroupWireFormat;openwire::marshal::v3::ProducerIdMars	
1313	2884
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller;openwire::marshal::v3::ProducerInfoMa	
1341	2921
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller;openwire::marshal::v3::RemoveInfoMars	
1372	3002
activemq::wireformat::openwire::marshal::v3::ContactGroupWireFormat;openwire::marshal::v3::RemoveSubscrip	
1400	3026
activemq::wireformat::openwire::marshal::v3::DataActiveResponseWireFormat;openwire::marshal::v3::ReplayCommand	
1433	3061
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller;openwire::marshal::v3::ResponseMarsha	
1494	3101
activemq::wireformat::openwire::marshal::v3::DestinationInfoWireFormat;openwire::marshal::v3::SessionIdMarsha	
1625	3183
activemq::wireformat::openwire::marshal::v3::DiscoveryInfoWireFormat;openwire::marshal::v3::SessionInfoMars	
1657	3206
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller;openwire::marshal::v3::ShutdownInfoMa	
1733	3271
activemq::wireformat::openwire::marshal::v3::FlushCommandWireFormat;openwire::marshal::v3::SubscriptionInfo	
1825	3441
activemq::wireformat::openwire::marshal::v3::IntegerResponseWireFormat;openwire::marshal::v3::TransactionIdMa	
1969	3586
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller;openwire::marshal::v3::TransactionInfo	
2031	3609
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller;openwire::marshal::v3::WireFormatInfo	
2056	3750
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller;openwire::marshal::v3::XATransactionI	

3788	1629
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	activemq::wireformat::openwire::marshal::v4::DiscoveryEventManager
181	1661
activemq::wireformat::openwire::marshal::v4::ActiveMQByteMessageMarshaller	activemq::wireformat::openwire::marshal::v4::ExceptionResponse
220	1741
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	activemq::wireformat::openwire::marshal::v4::FlushCommand
300	1829
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	activemq::wireformat::openwire::marshal::v4::IntegerResponse
339	1973
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	activemq::wireformat::openwire::marshal::v4::JournalQueueAck
366	2035
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	activemq::wireformat::openwire::marshal::v4::JournalTopicAck
410	2064
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	activemq::wireformat::openwire::marshal::v4::JournalTraceMap
452	2086
activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceMessageMarshaller	activemq::wireformat::openwire::marshal::v4::JournalTransaction
510	2117
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMap
536	2140
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	activemq::wireformat::openwire::marshal::v4::LastPartialCommand
563	2177
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	activemq::wireformat::openwire::marshal::v4::LocalTransaction
587	2223
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller
616	2414
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	activemq::wireformat::openwire::marshal::v4::MessageDispatch
643	2453
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	activemq::wireformat::openwire::marshal::v4::MessageDispatch
712	2481
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller
807	2506
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	activemq::wireformat::openwire::marshal::v4::MessageMarshaller
838	2538
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller
1183	2583
activemq::wireformat::openwire::marshal::v4::ConnectionFromMarshaller	activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilter
1215	2635
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	activemq::wireformat::openwire::marshal::v4::PartialCommand
1244	2751
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller
1274	2849
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller
1317	2880
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller
1344	2905
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller
1376	3014
activemq::wireformat::openwire::marshal::v4::ContentCommandMarshaller	activemq::wireformat::openwire::marshal::v4::RemoveSubscription
1404	3042
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller	activemq::wireformat::openwire::marshal::v4::ReplayCommand
1437	3049
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v4::ResponseMarshaller
1498	3088
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller

3172	1282
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	activemq::wireformat::openwire::marshal::v5::ConsumerControlResponseMarshaller
3214	1325
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller
3275	1352
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller
3452	1384
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller
3590	1412
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller
3617	1445
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller
3742	1482
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller
3780	1641
activemq::wireformat::openwire::marshal::v5::ActiveMQBlockWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::DiscoveryEventManager
189	1669
activemq::wireformat::openwire::marshal::v5::ActiveMQByteWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller
224	1737
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller
304	1837
activemq::wireformat::openwire::marshal::v5::ActiveMQMapWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller
343	1981
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller
370	2027
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller
414	2048
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller
456	2094
activemq::wireformat::openwire::marshal::v5::ActiveMQSequenceWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller
514	2113
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller
540	2144
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
567	2173
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	activemq::wireformat::openwire::marshal::v5::LocalTransactionMarshaller
595	2219
activemq::wireformat::openwire::marshal::v5::ActiveMQTextWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
624	2422
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
651	2449
activemq::wireformat::openwire::marshal::v5::BaseConsumerWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
725	2489
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller
815	2509
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	activemq::wireformat::openwire::marshal::v5::MessageMarshaller
846	2526
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller
1191	2575
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	activemq::wireformat::openwire::marshal::v5::NetworkBridgeFailureMarshaller
1223	2627
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller
1252	2742
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller

2857	655
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller
2888	732
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller
2917	819
activemq::wireformat::openwire::marshal::v5::RemoteInfoMarshaller	activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller
3010	850
activemq::wireformat::openwire::marshal::v5::RemoteSubscriptionInfoMarshaller	activemq::wireformat::openwire::marshal::v6::ConnectionContainerMarshaller
3038	1195
activemq::wireformat::openwire::marshal::v5::ReplyCommandMarshaller	activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller
3069	1227
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller
3097	1256
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller
3179	1286
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	activemq::wireformat::openwire::marshal::v6::ConsumerControllerMarshaller
3198	1329
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller
3267	1356
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller
3449	1388
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller	activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller
3575	1416
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller
3601	1449
activemq::wireformat::openwire::marshal::v5::WireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller
3730	1486
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller
3792	1637
activemq::wireformat::openwire::marshal::v6::ActiveMQBlockWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::DiscoveryEventManagerMarshaller
193	1650
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller	activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller
228	1725
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller	activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller
312	1817
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller
351	1961
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller
378	2020
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller
422	2060
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller	activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller
464	2082
activemq::wireformat::openwire::marshal::v6::ActiveMQSequenceMessageMarshaller	activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller
522	2101
activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller	activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller
547	2128
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller
575	2161
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	activemq::wireformat::openwire::marshal::v6::LocalTransactionMarshaller
603	2207
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller
628	2402
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller	activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller

2461	activemq::wireformat::openwire::marshal::BaseDataStreamMa
activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller,	755
2469	tightMarshalLong1
activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller,	756
2517	activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller2,
2546	activemq::wireformat::openwire::marshal::BaseDataStreamMa
activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller,	756
2591	tightMarshalNestedObject1
activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller,	757
2623	activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller,
2734	activemq::wireformat::openwire::marshal::v6::PrintMarshaller,
activemq::wireformat::openwire::marshal::v6::PrintMarshaller2	2710
2861	activemq::wireformat::openwire::marshal::BaseDataStreamMa
activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller,	757
2892	activemq::wireformat::openwire::OpenWireFormat,
activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller,	757
2925	tightMarshalObjectArray1
activemq::wireformat::openwire::marshal::v6::RemoveFromWireFormat,	757
2998	activemq::wireformat::openwire::marshal::v6::RemoveFromWireFormat2,
3034	activemq::wireformat::openwire::marshal::BaseDataStreamMa
activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller,	758
3065	tightMarshalString1
activemq::wireformat::openwire::marshal::v6::ResponseMarshaller,	758
3110	activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller2,
3175	activemq::wireformat::openwire::marshal::BaseDataStreamMa
activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller,	759
3194	tightUnmarshal
activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller,	759
3255	activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller,
3456	activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller,
3593	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM
activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller,	217
3613	activemq::wireformat::openwire::marshal::v1::ActiveMQBytes
activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller,	297
3734	activemq::wireformat::openwire::marshal::v1::ActiveMQDestin
activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller,	336
3772	activemq::wireformat::openwire::marshal::v1::ActiveMQMessa
tightMarshalBrokerError1	362
activemq::wireformat::openwire::marshal::BaseDataStr	754
754	activemq::wireformat::openwire::marshal::v1::ActiveMQObject
tightMarshalBrokerError2	406
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	448
755	activemq::wireformat::openwire::marshal::v1::ActiveMQQueue
tightMarshalCachedObject1	507
activemq::wireformat::openwire::marshal::BaseDataStr	533
755	activemq::wireformat::openwire::marshal::v1::ActiveMQTemp
tightMarshalCachedObject2	533

activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	2181
560	
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	2227
592	
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	2418
620	
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMessageMarshaller	2418
648	
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMessageMarshaller	2457
720	
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	2486
720	
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	2522
812	
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	2543
843	
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	2587
1188	
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2640
1219	
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	2755
1249	
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	2866
1279	
activemq::wireformat::openwire::marshal::v1::ConsumerGroupMarshaller	2897
1322	
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	2913
1349	
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	3006
1381	
activemq::wireformat::openwire::marshal::v1::ContactCommandMarshaller	3022
1408	
activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller	3053
1441	
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	3106
1503	
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	3188
1633	
activemq::wireformat::openwire::marshal::v1::DiscardResponseMarshaller	3203
1666	
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	3263
1746	
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3445
1834	
activemq::wireformat::openwire::marshal::v1::IntegrationResponseMarshaller	3579
1978	
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3606
2040	
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3747
2069	
activemq::wireformat::openwire::marshal::v1::JournalQueueMarshaller	3784
2091	
activemq::wireformat::openwire::marshal::v1::JournalQueueMarshaller	186
2122	
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	233
2148	

activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller;openwire::marshal::v2::FlushCommand	309	1822
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller;openwire::marshal::v2::IntegerResponse	348	1966
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller;openwire::marshal::v2::JournalQueueAck	374	2024
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller;openwire::marshal::v2::JournalTopicAck	418	2053
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller;openwire::marshal::v2::JournalTraceMa	460	2075
activemq::wireformat::openwire::marshal::v2::ActiveMQSequenceMessageMarshaller;openwire::marshal::v2::JournalTransact	519	2106
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller;openwire::marshal::v2::KeepAliveInfoM	544	2132
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller;openwire::marshal::v2::LastPartialComm	572	2169
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller;openwire::marshal::v2::LocalTransaction	600	2211
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller;openwire::marshal::v2::MessageAckMar	632	2406
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller;openwire::marshal::v2::MessageDispatch	660	2441
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller;openwire::marshal::v2::MessageDispatch	740	2474
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller;openwire::marshal::v2::MessageIdMarsh	824	2502
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller;openwire::marshal::v2::MessageMarsh	855	2535
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller;openwire::marshal::v2::MessagePullMar	1200	2571
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller;openwire::marshal::v2::NetworkBridgeF	1207	2620
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller;openwire::marshal::v2::PartialCommand	1237	2739
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller;openwire::marshal::v2::ProducerAckMar	1267	2846
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller;openwire::marshal::v2::ProducerIdMars	1310	2877
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller;openwire::marshal::v2::ProducerInfoMa	1337	2909
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller;openwire::marshal::v2::RemoveInfoMars	1369	2994
activemq::wireformat::openwire::marshal::v2::ContentCommandMarshaller;openwire::marshal::v2::RemoveSubscrip	1396	3030
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller;openwire::marshal::v2::ReplayCommand	1429	3057
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller;openwire::marshal::v2::ResponseMarsha	1491	3093
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller;openwire::marshal::v2::SessionIdMarsha	1621	3168
activemq::wireformat::openwire::marshal::v2::DiscardRequestMarshaller;openwire::marshal::v2::SessionInfoMars	1654	3211
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller;openwire::marshal::v2::ShutdownInfoM	1730	3259

activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	1373
3461	
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller	1400
3583	
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	1433
3622	
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	1495
3739	
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	1625
3776	
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller	1658
174	
activemq::wireformat::openwire::marshal::v3::ActiveMQByteMessageMarshaller	1734
213	
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	1826
293	
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	1970
332	
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	2032
358	
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	2057
402	
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMessageMarshaller	2079
444	
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamingMessageMarshaller	2110
503	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	2136
530	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	2165
556	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	2215
584	
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	2410
612	
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	2445
640	
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	2478
707	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2514
804	
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	2531
835	
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2579
1180	
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2632
1211	
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2747
1241	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	2854
1271	
activemq::wireformat::openwire::marshal::v3::ConsumerGroupMarshaller	2885
1314	
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	2921
1341	
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	

activemq::wireformat::openwire::marshal::v3::RemoteInfoMarshaller	3002	activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	839
activemq::wireformat::openwire::marshal::v3::RemoteSubscriptionInfoMarshaller	3026	activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	1184
activemq::wireformat::openwire::marshal::v3::ReplyCommandMarshaller	3061	activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	1215
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	3102	activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	1245
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	3184	activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1275
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	3207	activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	1318
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	3271	activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	1345
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	3441	activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1377
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	3587	activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller	1404
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	3610	activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller	1437
activemq::wireformat::openwire::marshal::v3::WireFormatMarshaller	3751	activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	1499
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	3788	activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	1629
activemq::wireformat::openwire::marshal::v4::ActiveMQBlockWireFormatMarshaller	182	activemq::wireformat::openwire::marshal::v4::DiscoveryEventManager	1662
activemq::wireformat::openwire::marshal::v4::ActiveMQByteMessageMarshaller	221	activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	1742
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	301	activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	1830
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	340	activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	1974
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	366	activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	2036
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	410	activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	2065
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	452	activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller	2087
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	511	activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	2118
activemq::wireformat::openwire::marshal::v4::ActiveMQReplyDestinationMarshaller	537	activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	2140
activemq::wireformat::openwire::marshal::v4::ActiveMQReplyQueueMarshaller	564	activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	2177
activemq::wireformat::openwire::marshal::v4::ActiveMQReplyTopicMarshaller	588	activemq::wireformat::openwire::marshal::v4::LocalTransactionMarshaller	2223
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	616	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	2414
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	644	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2453
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	713	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2482
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	808	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	2506

activemq::wireformat::openwire::marshal::v4::MessageMarshaller	541
2539	
activemq::wireformat::openwire::marshal::v4::MessagePublicWireFormat	568
2583	
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller	596
2636	
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	624
2751	
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	652
2850	
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	727
2881	
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	816
2905	
activemq::wireformat::openwire::marshal::v4::RemoteInfoMarshaller	847
3014	
activemq::wireformat::openwire::marshal::v4::RemoteSubscriptionInfoMarshaller	1192
3042	
activemq::wireformat::openwire::marshal::v4::ReplaceCommandMarshaller	1223
3049	
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	1253
3088	
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	1283
3172	
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	1326
3215	
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	1353
3275	
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	1385
3453	
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	1412
3590	
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	1445
3618	
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	1483
3743	
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	1641
3780	
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobWireFormatMarshaller	1670
190	
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	1738
225	
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	1838
305	
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	1982
344	
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	2028
370	
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectWireMessageMarshaller	2049
414	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	2095
456	
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	2114
515	

activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller	2144	activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessage	229
activemq::wireformat::openwire::marshal::v5::LastReceivedCommandMarshaller	2173	activemq::wireformat::openwire::marshal::v6::ActiveMQDestination	313
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller	2219	activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessage	352
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller	2422	activemq::wireformat::openwire::marshal::v6::ActiveMQMessage	378
activemq::wireformat::openwire::marshal::v5::MessageDispatchInfoMarshaller	2449	activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessage	422
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	2490	activemq::wireformat::openwire::marshal::v6::ActiveMQQueue	464
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller	2510	activemq::wireformat::openwire::marshal::v6::ActiveMQStream	523
activemq::wireformat::openwire::marshal::v5::MessageMarshaller	2527	activemq::wireformat::openwire::marshal::v6::ActiveMQTemplate	548
activemq::wireformat::openwire::marshal::v5::MessagePullInfoMarshaller	2575	activemq::wireformat::openwire::marshal::v6::ActiveMQTemporaryQueue	576
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller	2628	activemq::wireformat::openwire::marshal::v6::ActiveMQTemporaryQueue	604
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	2743	activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessage	628
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	2858	activemq::wireformat::openwire::marshal::v6::ActiveMQTopic	656
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	2889	activemq::wireformat::openwire::marshal::v6::BaseCommandMessage	733
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	2917	activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	820
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller	3010	activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller	851
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	3038	activemq::wireformat::openwire::marshal::v6::ConnectionContainer	1196
activemq::wireformat::openwire::marshal::v5::ReplyCommandMarshaller	3069	activemq::wireformat::openwire::marshal::v6::ConnectionError	1227
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	3097	activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller	1257
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	3180	activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller	1287
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	3199	activemq::wireformat::openwire::marshal::v6::ConsumerController	1330
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	3267	activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	1357
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	3449	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller	1389
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller	3576	activemq::wireformat::openwire::marshal::v6::ControlCommand	1416
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	3602	activemq::wireformat::openwire::marshal::v6::DataArrayResponse	1449
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	3731	activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller	1487
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	3792	activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller	1637
activemq::wireformat::openwire::marshal::v6::ActiveMQBlockReferenceMarshaller	194	activemq::wireformat::openwire::marshal::v6::DiscoveryEventManager	1650

Generated on Sun Sep 11 2011 18:23:35 for activemq-cpp-3.2.1 by Doxygen

- decaf::util::TimerTask, 3556
- TimerImpl
 - decaf::util::TimerTask, 3556
- TimerTask
 - decaf::util::TimerTask, 3555
- TimerTaskHeap
 - decaf::internal::util::TimerTaskHeap, 3557
- timestamp
 - activemq::commands::Message, 2374
 - decaf::util::UUID, 3711
- TimeUnit
 - decaf::util::concurrent::TimeUnit, 3561
- toArray
 - activemq::util::ActiveMQProperties, 434
 - cms::CMSProperties, 1082
 - decaf::util::AbstractCollection, 153
 - decaf::util::Collection, 1107
 - decaf::util::concurrent::SynchronousQueue, 3486
 - decaf::util::Properties, 2935
 - decaf::util::StlQueue, 3387
 - decaf::util::StringTokenizer, 3433
- toBinaryString
 - decaf::lang::Integer, 1952
 - decaf::lang::Long, 2278
- toByteArray
 - decaf::io::ByteArrayOutputStream, 953
- toDays
 - decaf::util::concurrent::TimeUnit, 3564
- toDegrees
 - decaf::lang::Math, 2354
- toDestinationOption
 - activemq::core::ActiveMQConstants, 268
- toHexFromBytes
 - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 762
- toHexString
 - decaf::lang::Double, 1680
 - decaf::lang::Float, 1788
 - decaf::lang::Integer, 1953
 - decaf::lang::Long, 2279
- toHours
 - decaf::util::concurrent::TimeUnit, 3565
- toMicros
 - decaf::util::concurrent::TimeUnit, 3565
- toMillis
 - decaf::util::concurrent::TimeUnit, 3565
- toMinutes
 - decaf::util::concurrent::TimeUnit, 3566
- toNanos
 - decaf::util::concurrent::TimeUnit, 3566
- toOctalString
 - decaf::lang::Integer, 1953
 - decaf::lang::Long, 2279
- TOPIC
 - cms::Destination, 1611
- TOPIC_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 3397
- TOPIC_QUALIFIED_PREFIX
 - activemq::commands::ActiveMQDestination, 289
- toRadians
 - decaf::lang::Math, 2354
- toSeconds
 - decaf::util::concurrent::TimeUnit, 3566
- toStream
 - activemq::wireformat::stomp::StompFrame, 3402
- toString
 - activemq::commands::ActiveMQBlobMessage, 170
 - activemq::commands::ActiveMQBytesMessage, 205
 - activemq::commands::ActiveMQDestination, 287
 - activemq::commands::ActiveMQMapMessage, 328
 - activemq::commands::ActiveMQMessage, 354
 - activemq::commands::ActiveMQObjectMessage, 398
 - activemq::commands::ActiveMQQueue, 438
 - activemq::commands::ActiveMQStreamMessage, 495
 - activemq::commands::ActiveMQTempDestination, 526
 - activemq::commands::ActiveMQTempQueue, 552
 - activemq::commands::ActiveMQTempTopic, 580
 - activemq::commands::ActiveMQTextMessage, 608
 - activemq::commands::ActiveMQTopic, 636
 - activemq::commands::BaseCommand, 700
 - activemq::commands::BaseDataStructure, 767
 - activemq::commands::BooleanExpression, 787
 - activemq::commands::BrokerId, 800
 - activemq::commands::BrokerInfo, 829
 - activemq::commands::Command, 1111
 - activemq::commands::ConnectionControl, 1175
 - activemq::commands::ConnectionError, 1203
 - activemq::commands::ConnectionId, 1233

- activemq::commands::ConnectionInfo, 1262
- activemq::commands::ConsumerControl, 1305
- activemq::commands::ConsumerId, 1333
- activemq::commands::ConsumerInfo, 1363
- activemq::commands::ControlCommand, 1392
- activemq::commands::DataArrayResponse, 1425
- activemq::commands::DataResponse, 1479
- activemq::commands::DataStructure, 1558
- activemq::commands::DestinationInfo, 1617
- activemq::commands::DiscoveryEvent, 1646
- activemq::commands::ExceptionResponse, 1722
- activemq::commands::FlushCommand, 1814
- activemq::commands::IntegerResponse, 1958
- activemq::commands::JournalQueueAck, 2016
- activemq::commands::JournalTopicAck, 2044
- activemq::commands::JournalTrace, 2071
- activemq::commands::JournalTransaction, 2098
- activemq::commands::KeepAliveInfo, 2124
- activemq::commands::LastPartialCommand, 2158
- activemq::commands::LocalTransactionId, 2203
- activemq::commands::Message, 2372
- activemq::commands::MessageAck, 2398
- activemq::commands::MessageDispatch, 2430
- activemq::commands::MessageDispatchNotification, 2465
- activemq::commands::MessageId, 2498
- activemq::commands::MessagePull, 2567
- activemq::commands::NetworkBridgeFilter, 2616
- activemq::commands::PartialCommand, 2730
- activemq::commands::ProducerAck, 2842
- activemq::commands::ProducerId, 2873
- activemq::commands::ProducerInfo, 2901
- activemq::commands::RemoveInfo, 2990
- activemq::commands::RemoveSubscriptionInfo, 3018
- activemq::commands::ReplayCommand, 3045
- activemq::commands::Response, 3079
- activemq::commands::SessionId, 3164
- activemq::commands::SessionInfo, 3191
- activemq::commands::ShutdownInfo, 3251
- activemq::commands::SubscriptionInfo, 3437
- activemq::commands::TransactionId, 3572
- activemq::commands::TransactionInfo, 3597
- activemq::commands::WireFormatInfo, 3727
- activemq::commands::XATransactionId, 3768
- activemq::core::ActiveMQConstants, 268
- activemq::state::ConnectionState, 1293
- activemq::state::ConsumerState, 1390
- activemq::state::ProducerState, 2926
- activemq::state::SessionState, 3219
- activemq::state::TransactionState, 3624
- activemq::util::ActiveMQProperties, 434
- activemq::util::PrimitiveList, 2798
- activemq::util::PrimitiveMap, 2807
- activemq::util::PrimitiveValueNode, 2830
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 763
- cms::CMSProperties, 1082
- decaf::io::ByteArrayOutputStream, 953
- decaf::io::FilterOutputStream, 1779
- decaf::io::InputStream, 1917
- decaf::io::OutputStream, 2722
- decaf::lang::Boolean, 784, 785
- decaf::lang::Byte, 891
- decaf::lang::Character, 1026
- decaf::lang::CharSequence, 1055
- decaf::lang::Double, 1681
- decaf::lang::Float, 1789
- decaf::lang::Integer, 1954
- decaf::lang::Long, 2279, 2280
- decaf::lang::Short, 3227
- decaf::lang::String, 3430
- decaf::lang::Thread, 3529
- decaf::net::InetAddress, 1890
- decaf::net::ServerSocket, 3145
- decaf::net::Socket, 3297
- decaf::net::SocketImpl, 3313
- decaf::net::URI, 3671
- decaf::nio::ByteBuffer, 978
- decaf::nio::CharBuffer, 1052
- decaf::nio::DoubleBuffer, 1702
- decaf::nio::FloatBuffer, 1810
- decaf::nio::IntBuffer, 1940
- decaf::nio::LongBuffer, 2301
- decaf::nio::ShortBuffer, 3248
- decaf::security::cert::Certificate, 1009

- decaf::util::concurrent::atomic::AtomicBoolean, 679
- decaf::util::concurrent::atomic::AtomicInteger, 684
- decaf::util::concurrent::atomic::AtomicReference, 689
- decaf::util::concurrent::locks::ReentrantLock, 2982
- decaf::util::concurrent::Semaphore, 3132
- decaf::util::concurrent::TimeUnit, 3567
- decaf::util::Date, 1562
- decaf::util::logging::Level, 2188
- decaf::util::Properties, 2935
- decaf::util::UUID, 3711
- total
 - inflate_state, 1893
- total_in
 - z_stream_s, 3795
- total_out
 - z_stream_s, 3795
- toURI
 - activemq::util::CompositeData, 1131
- toURIOption
 - activemq::core::ActiveMQConstants, 268
- toURL
 - decaf::net::URI, 3671
- Trace
 - zutil.h, 4217
- Tracec
 - zutil.h, 4217
- Tracecv
 - zutil.h, 4217
- Tracev
 - zutil.h, 4217
- Tracevv
 - zutil.h, 4217
- track
 - activemq::state::ConnectionStateTracker, 1300
- trackBack
 - activemq::state::ConnectionStateTracker, 1300
- Tracked
 - activemq::state::Tracked, 3569
- TRANSACTION_STATE_BEGIN
 - activemq::core::ActiveMQConstants, 267
- TRANSACTION_STATE_-
 - COMMITONEPHASE
 - activemq::core::ActiveMQConstants, 267
 - COMMITTWOPHASE
 - activemq::core::ActiveMQConstants, 267
- TRANSACTION_STATE_END
 - activemq::core::ActiveMQConstants, 267
- TRANSACTION_STATE_FORGET
 - activemq::core::ActiveMQConstants, 267
- TRANSACTION_STATE_PREPARE
 - activemq::core::ActiveMQConstants, 267
- TRANSACTION_STATE_RECOVER
 - activemq::core::ActiveMQConstants, 267
- TRANSACTION_STATE_ROLLBACK
 - activemq::core::ActiveMQConstants, 267
- TransactionId
 - activemq::commands::TransactionId, 3571
- transactionId
 - activemq::commands::JournalTopicAck, 2045
 - activemq::commands::JournalTransaction, 2098
 - activemq::commands::Message, 2374
 - activemq::commands::MessageAck, 2399
 - activemq::commands::TransactionInfo, 3598
- TransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::TransactionIdMa, 3577
 - activemq::wireformat::openwire::marshal::v2::TransactionIdMa, 3581
 - activemq::wireformat::openwire::marshal::v3::TransactionIdMa, 3585
 - activemq::wireformat::openwire::marshal::v4::TransactionIdMa, 3588
 - activemq::wireformat::openwire::marshal::v5::TransactionIdMa, 3574
 - activemq::wireformat::openwire::marshal::v6::TransactionIdMa, 3592
- TransactionInfo
 - activemq::commands::TransactionInfo, 3596
- TransactionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::TransactionInfo, 3604
 - activemq::wireformat::openwire::marshal::v2::TransactionInfo, 3620
 - activemq::wireformat::openwire::marshal::v3::TransactionInfo, 3608
 - activemq::wireformat::openwire::marshal::v4::TransactionInfo, 3616
 - activemq::wireformat::openwire::marshal::v5::TransactionInfo, 3600
 - activemq::wireformat::openwire::marshal::v6::TransactionInfo, 3612
- TransactionState
 - activemq::core::ActiveMQConstants, 267
 - activemq::state::TransactionState, 3624
- transfer
 - decaf::internal::util::concurrent::TransferQueue, 3626

- decaf::internal::util::concurrent::TransferStack, 3628
- TransferQueue
 - decaf::internal::util::concurrent::TransferQueue, 3626
- TransferStack
 - decaf::internal::util::concurrent::TransferStack, 3628
- TransportFilter
 - activemq::transport::TransportFilter, 3638
- transportInterrupted
 - activemq::core::ActiveMQConnection, 251
 - activemq::state::ConnectionStateTracker, 1300
- activemq::transport::DefaultTransportListener, 1594
- activemq::transport::failover::FailoverTransportListener, 1767
- activemq::transport::TransportFilter, 3643
- activemq::transport::TransportListener, 3645
- transportResumed
 - activemq::core::ActiveMQConnection, 251
 - activemq::transport::DefaultTransportListener, 1594
 - activemq::transport::failover::FailoverTransportListener, 1767
 - activemq::transport::TransportFilter, 3643
 - activemq::transport::TransportListener, 3645
- tree_desc
 - deflate.h, 4202
- tree_desc_s, 3648
 - dyn_tree, 3648
 - max_code, 3648
 - stat_desc, 3648
- trees.h
 - _dist_code, 4207
 - _length_code, 4208
 - base_dist, 4208
 - base_length, 4208
 - static_dtree, 4209
 - static_ltree, 4209
- TRY_FREE
 - zutil.h, 4217
- tryAcquire
 - decaf::util::concurrent::Semaphore, 3132–3134
- tryLock
 - activemq::core::MessageDispatchChannel, 2436
 - decaf::internal::util::concurrent::SynchronizableLock, 3475
 - decaf::io::InputStream, 1917
 - decaf::io::OutputStream, 2722
 - decaf::util::AbstractCollection, 153
 - decaf::util::concurrent::ConcurrentStlMap, 1153
 - decaf::util::concurrent::locks::Lock, 2232, 2233
 - decaf::util::concurrent::locks::ReentrantLock, 2982, 2983
 - decaf::util::concurrent::Mutex, 2606
 - decaf::util::concurrent::Synchronizable, 3466
 - decaf::util::StlMap, 3379
 - decaf::util::StlQueue, 3388
 - trylock
 - decaf::internal::util::concurrent::MutexImpl, 2610
 - inflate.h, 4206
 - type
 - activemq::commands::JournalTransaction, 2098
 - activemq::commands::Message, 2374
 - activemq::commands::TransactionInfo, 3598
- TYPEDO
 - inflate.h, 4206
- uch
 - zutil.h, 4217
- uchf
 - zutil.h, 4217
- uInt
 - zconf.h, 4211
- uIntf
 - zconf.h, 4211
- ulg
 - zutil.h, 4217
- uLong
 - zconf.h, 4211
- uLongf
 - zconf.h, 4211
- uncaughtException
 - decaf::lang::Thread::UncaughtExceptionHandler, 3649
- UnknownHostException
 - decaf::net::UnknownHostException, 3650, 3651
- UnknownServiceException
 - decaf::net::UnknownServiceException, 3652, 3653
 - activemq::core::MessageDispatchChannel, 2436

- decaf::internal::util::concurrent::MutexImpl, unsetenv 2611
- decaf::internal::util::concurrent::Synchronizable, UNSUBSCRIBE 3475
- decaf::io::InputStream, 1918
- decaf::io::OutputStream, 2723
- decaf::util::AbstractCollection, 154
- decaf::util::concurrent::ConcurrentStlMap, 1154
- decaf::util::concurrent::Lock, 2229
- decaf::util::concurrent::locks::Lock, 2234
- decaf::util::concurrent::locks::ReentrantLock, 2984
- decaf::util::concurrent::Mutex, 2607
- decaf::util::concurrent::Synchronizable, 3467
- decaf::util::StlMap, 3380
- decaf::util::StlQueue, 3388
- unmarshal
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshalChecksum, 1060, 1061 2812
 - activemq::wireformat::openwire::OpenWireFormat, 2711
 - activemq::wireformat::openwire::utils::BooleanStlEncoder, 789
 - activemq::wireformat::stomp::StompWireFormat, 3409
 - activemq::wireformat::WireFormat, 3715
- unmarshalList
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshalChecksum, 1060, 1061 2813
- unmarshalMap
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshalChecksum, 1060, 1061 2813
- unmarshalPrimitive
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshalChecksum, 1060, 1061 2813
- unmarshalPrimitiveList
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshalChecksum, 1060, 1061 2814
- unmarshalPrimitiveMap
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshalChecksum, 1060, 1061 2814
- unpark
 - decaf::util::concurrent::locks::LockSupport, 2237
- unread
 - decaf::io::PushbackInputStream, 2945, 2946
- unregisterFactory
 - activemq::transport::TransportRegistry, 3647
 - activemq::wireformat::WireFormatRegistry, 3754
- decaf::lang::System, 3494
- activemq::wireformat::stomp::StompCommandConstants, 3397
- unsubscribe
 - activemq::cmsutil::PooledSession, 2776
 - activemq::core::ActiveMQSession, 481
 - cms::Session, 3160
- UnsupportedEncodingException
 - decaf::io::UnsupportedEncodingException, 3655, 3656
- UnsupportedOperationException
 - cms::UnsupportedOperationException, 3660
 - decaf::lang::exceptions::UnsupportedOperationException, 3657, 3658
- update
 - decaf::util::zip::Adler32, 664, 665
 - decaf::util::zip::CRC32, 1421, 1422
 - decaf::net::URI, 3663, 3664
 - decaf::internal::net::URIEncoderDecoder, 3673
 - URIHelper
 - decaf::internal::net::URIHelper, 3676
 - URIParam
 - activemq::core::ActiveMQConstants::StaticInitializer, 3681
 - uriParams
 - activemq::core::ActiveMQConstants::StaticInitializer, 3681
 - URIPool
 - activemq::transport::failover::URIPool, 3682
 - URISyntaxException
 - decaf::net::URISyntaxException, 3687–3689
 - URIType
 - decaf::internal::net::URIType, 3692
 - URL
 - decaf::net::URL, 3699
 - userID
 - activemq::commands::Message, 2374
 - userName
 - activemq::commands::ConnectionInfo, 1263
 - ush
 - zutil.h, 4217
 - ushf

- zutil.h, 4217
- UTFDataFormatException
 - decaf::io::UTFDataFormatException, 3704, 3705
- UUID
 - decaf::util::UUID, 3708
- val
 - code, 1097
- valid
 - decaf::io::FileDescriptor, 1769
- validate
 - decaf::internal::net::URLEncoderDecoder, 3674
- validateAuthority
 - decaf::internal::net::URIHelper, 3679
- validateFragment
 - decaf::internal::net::URIHelper, 3679
- validatePath
 - decaf::internal::net::URIHelper, 3679
- validateQuery
 - decaf::internal::net::URIHelper, 3680
- validateScheme
 - decaf::internal::net::URIHelper, 3680
- validateSimple
 - decaf::internal::net::URLEncoderDecoder, 3674
- validateSsp
 - decaf::internal::net::URIHelper, 3680
- validateUserInfo
 - decaf::internal::net::URIHelper, 3681
- value
 - activemq::commands::BrokerId, 801
 - activemq::commands::ConnectionId, 1234
 - activemq::commands::ConsumerId, 1334
 - activemq::commands::LocalTransactionId, 2204
 - activemq::commands::ProducerId, 2874
 - activemq::commands::SessionId, 3165
- valueOf
 - decaf::lang::Boolean, 785
 - decaf::lang::Byte, 891, 892
 - decaf::lang::Character, 1026
 - decaf::lang::Double, 1682
 - decaf::lang::Float, 1790
 - decaf::lang::Integer, 1954, 1955
 - decaf::lang::Long, 2280, 2281
 - decaf::lang::Short, 3228
 - decaf::util::concurrent::TimeUnit, 3567
- values
 - decaf::util::concurrent::ConcurrentStlMap, 1154
 - decaf::util::concurrent::TimeUnit, 3568
 - decaf::util::Map, 2317
- decaf::util::StlMap, 3380
- variant
 - decaf::util::UUID, 3711
- verify
 - decaf::security::cert::Certificate, 1009
- version
 - decaf::util::UUID, 3712
- visit
 - activemq::commands::BrokerError, 796
 - activemq::commands::BrokerInfo, 830
 - activemq::commands::Command, 1112
 - activemq::commands::ConnectionControl, 1176
 - activemq::commands::ConnectionError, 1203
 - activemq::commands::ConnectionInfo, 1262
 - activemq::commands::ConsumerControl, 1306
 - activemq::commands::ConsumerInfo, 1364
 - activemq::commands::ControlCommand, 1392
 - activemq::commands::DestinationInfo, 1617
 - activemq::commands::FlushCommand, 1814
 - activemq::commands::KeepAliveInfo, 2124
 - activemq::commands::Message, 2373
 - activemq::commands::MessageAck, 2398
 - activemq::commands::MessageDispatch, 2430
 - activemq::commands::MessageDispatchNotification, 2466
 - activemq::commands::MessagePull, 2567
 - activemq::commands::ProducerAck, 2842
 - activemq::commands::ProducerInfo, 2901
 - activemq::commands::RemoveInfo, 2990
 - activemq::commands::RemoveSubscriptionInfo, 3018
 - activemq::commands::ReplayCommand, 3045
 - activemq::commands::Response, 3079
 - activemq::commands::SessionInfo, 3191
 - activemq::commands::ShutdownInfo, 3251
 - activemq::commands::TransactionInfo, 3598
 - activemq::commands::WireFormatInfo, 3727
- voidp
 - zconf.h, 4211
- voidpc
 - zconf.h, 4211
- voidpf
 - zconf.h, 4211

- w_bits
 - internal_state, 1985
- w_mask
 - internal_state, 1985
- w_size
 - internal_state, 1985
- wait
 - activemq::core::MessageDispatchChannel, 2436, 2437
 - decaf::internal::util::concurrent::ConditionImpl, 1165
 - decaf::internal::util::concurrent::SynchronizableImpl, 3475, 3476
 - decaf::io::InputStream, 1918
 - decaf::io::OutputStream, 2723, 2724
 - decaf::util::AbstractCollection, 154, 155
 - decaf::util::concurrent::ConcurrentStlMap, 1154, 1155
 - decaf::util::concurrent::Mutex, 2607, 2608
 - decaf::util::concurrent::Synchronizable, 3468, 3470, 3471
 - decaf::util::StlMap, 3380, 3381
 - decaf::util::StlQueue, 3388, 3389
- WAIT_INFINITE
 - Concurrent.h, 4287
- waitForSpace
 - activemq::util::MemoryUsage, 2357
 - activemq::util::Usage, 3703
- WAITING
 - decaf::lang::Thread, 3523
- wakeup
 - activemq::core::ActiveMQSession, 481
 - activemq::core::ActiveMQSessionExecutor, 485
 - activemq::threads::CompositeTaskRunner, 1134
 - activemq::threads::DedicatedTaskRunner, 1565
 - activemq::threads::TaskRunner, 3497
- want
 - gz_state, 1851
- Warn
 - decaf::util::logging, 140
- warn
 - decaf::util::logging::SimpleLogger, 3281
- WARNING
 - decaf::util::logging::Level, 2190
- warning
 - decaf::util::logging::Logger, 2249
- was
 - inflate_state, 1893
- wasPrepared
 - activemq::commands::JournalTransaction, 2098
- wbits
 - inflate_state, 1893
- what
 - cms::CMSException, 1077
 - decaf::lang::Exception, 1718
- whave
 - inflate_state, 1893
- WIN_INIT
 - deflate.h, 4202
- Window
 - inflate_state, 1893
 - internal_state, 1985
- window_size
 - internal_state, 1985
- windowSize
 - activemq::commands::ProducerInfo, 2902
- WireFormatInfo
 - activemq::commands::WireFormatInfo, 3720
- WireFormatInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::WireFormatInfo, 3745
 - activemq::wireformat::openwire::marshal::v2::WireFormatInfo, 3737
 - activemq::wireformat::openwire::marshal::v3::WireFormatInfo, 3749
 - activemq::wireformat::openwire::marshal::v4::WireFormatInfo, 3741
 - activemq::wireformat::openwire::marshal::v5::WireFormatInfo, 3729
 - activemq::wireformat::openwire::marshal::v6::WireFormatInfo, 3733
- WireFormatNegotiator
 - activemq::wireformat::WireFormatNegotiator, 3752
- wnext
 - inflate_state, 1893
- work
 - inflate_state, 1893
- wrap
 - decaf::nio::ByteBuffer, 978
 - decaf::nio::CharBuffer, 1052
 - decaf::nio::DoubleBuffer, 1702, 1703
 - decaf::nio::FloatBuffer, 1810
 - decaf::nio::IntBuffer, 1940, 1941
 - decaf::nio::LongBuffer, 2301
 - decaf::nio::ShortBuffer, 3248, 3249
 - inflate_state, 1893
 - internal_state, 1985
- write
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2685
 - decaf::internal::net::tcp::TcpSocket, 3505

- decaf::internal::util::ByteArrayAdapter, 914
- decaf::io::OutputStream, 2724, 2725
- decaf::io::Writer, 3759–3761
- WRITE_FAILURE
 - decaf::util::logging::ErrorManager, 1711
- writeBoolean
 - activemq::commands::ActiveMQBytesMessage, 206
 - activemq::commands::ActiveMQStreamMessage, 495
 - activemq::wireformat::openwire::utils::BooleanStream, 790
 - cms::BytesMessage, 988
 - cms::StreamMessage, 3423
 - decaf::io::DataOutput, 1470
 - decaf::io::DataOutputStream, 1475
- writeByte
 - activemq::commands::ActiveMQBytesMessage, 206
 - activemq::commands::ActiveMQStreamMessage, 496
 - cms::BytesMessage, 989
 - cms::StreamMessage, 3423
 - decaf::io::DataOutput, 1470
 - decaf::io::DataOutputStream, 1476
- writeBytes
 - activemq::commands::ActiveMQBytesMessage, 206
 - activemq::commands::ActiveMQStreamMessage, 496
 - cms::BytesMessage, 989
 - cms::StreamMessage, 3424
 - decaf::io::DataOutput, 1470
 - decaf::io::DataOutputStream, 1476
- writeChar
 - activemq::commands::ActiveMQBytesMessage, 207
 - activemq::commands::ActiveMQStreamMessage, 497
 - cms::BytesMessage, 990
 - cms::StreamMessage, 3424
 - decaf::io::DataOutput, 1470
 - decaf::io::DataOutputStream, 1476
- writeChars
 - decaf::io::DataOutput, 1471
 - decaf::io::DataOutputStream, 1476
- WriteChecker
 - activemq::transport::inactivity::InactivityMonitor, 1877
 - activemq::transport::inactivity::WriteChecker, 3755
- writeDouble
 - activemq::commands::ActiveMQBytesMessage, 207
 - activemq::commands::ActiveMQStreamMessage, 497
 - cms::BytesMessage, 990
 - cms::StreamMessage, 3425
 - decaf::io::DataOutput, 1471
 - decaf::io::DataOutputStream, 1476
- writeFloat
 - activemq::commands::ActiveMQBytesMessage, 207
 - activemq::commands::ActiveMQStreamMessage, 497
 - cms::BytesMessage, 990
 - cms::StreamMessage, 3425
 - decaf::io::DataOutput, 1471
 - decaf::io::DataOutputStream, 1476
- writeInt
 - activemq::commands::ActiveMQBytesMessage, 208
 - activemq::commands::ActiveMQStreamMessage, 498
 - cms::BytesMessage, 991
 - cms::StreamMessage, 3425
 - decaf::io::DataOutput, 1472
 - decaf::io::DataOutputStream, 1476
- writeLock
 - decaf::util::concurrent::locks::ReadWriteLock, 2970
- writeLong
 - activemq::commands::ActiveMQBytesMessage, 208
 - activemq::commands::ActiveMQStreamMessage, 498
 - cms::BytesMessage, 991
 - cms::StreamMessage, 3426
 - decaf::io::DataOutput, 1472
 - decaf::io::DataOutputStream, 1476
- Writer
 - decaf::io::Writer, 3757
- writeShort
 - activemq::commands::ActiveMQBytesMessage, 208
 - activemq::commands::ActiveMQStreamMessage, 498
 - cms::BytesMessage, 991
 - cms::StreamMessage, 3426
 - decaf::io::DataOutput, 1472
 - decaf::io::DataOutputStream, 1476
- writeString
 - activemq::commands::ActiveMQBytesMessage, 208
 - activemq::commands::ActiveMQStreamMessage, 498

- activemq::util::MarshallingSupport, 2338
- cms::BytesMessage, 992
- cms::StreamMessage, 3426
- writeString16
 - activemq::util::MarshallingSupport, 2338
- writeString32
 - activemq::util::MarshallingSupport, 2338
- writeTo
 - decaf::io::ByteArrayOutputStream, 953
- writeUnsignedShort
 - activemq::commands::ActiveMQBytesMessage, 209
 - activemq::commands::ActiveMQStreamMessage, 499
 - cms::BytesMessage, 992
 - cms::StreamMessage, 3427
 - decaf::io::DataOutput, 1472
 - decaf::io::DataOutputStream, 1476
- writeUTF
 - activemq::commands::ActiveMQBytesMessage, 209
 - cms::BytesMessage, 992
 - decaf::io::DataOutput, 1473
 - decaf::io::DataOutputStream, 1476
- written
 - decaf::io::DataOutputStream, 1476
- wsize
 - inflate_state, 1893
- XATransactionId
 - activemq::commands::XATransactionId, 3766
- XATransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 3782
 - activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 3774
 - activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 3786
 - activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 3778
 - activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 3790
 - activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 3770
- xflags
 - gz_header_s, 1849
- XMLFormatter
 - decaf::util::logging::XMLFormatter, 3794
- yield
 - decaf::lang::Thread, 3529
- Z_ASCII
 - zlib.h, 4215
- Z_BEST_COMPRESSION
 - zlib.h, 4215
- Z_BEST_SPEED
 - zlib.h, 4215
- Z_BINARY
 - zlib.h, 4215
- Z_BLOCK
 - zlib.h, 4215
- Z_BUF_ERROR
 - zlib.h, 4215
- Z_DATA_ERROR
 - zlib.h, 4215
- Z_DEFAULT_COMPRESSION
 - zlib.h, 4215
- Z_DEFAULT_STRATEGY
 - zlib.h, 4215
- Z_DEFLATED
 - zlib.h, 4215
- Z_ERRMSG
 - zutil.h, 4217
- Z_ERRNO
 - zlib.h, 4215
- Z_FILTERED
 - zlib.h, 4215
- Z_FINISH
 - zlib.h, 4215
- Z_FIXED
 - zlib.h, 4215
- Z_FULL_FLUSH
 - zlib.h, 4215
- Z_HUFFMAN_ONLY
 - zlib.h, 4215
- Z_MEM_ERROR
 - zlib.h, 4215
- Z_NEED_DICT
 - zlib.h, 4215
- Z_NO_COMPRESSION
 - zlib.h, 4215
- Z_NO_FLUSH
 - zlib.h, 4215
- Z_NULL
 - zlib.h, 4215
- Z_OFF64
 - zconf.h, 4211
- z_off_t
 - zconf.h, 4211
- Z_OK
 - zlib.h, 4215
- Z_PARTIAL_FLUSH
 - zlib.h, 4215
- Z_RLE
 - zlib.h, 4215
- z_stream
 - zlib.h, 4215

- zlib.h, 4215
- Z_STREAM_END
 - zlib.h, 4215
- Z_STREAM_ERROR
 - zlib.h, 4215
- z_stream_s, 3795
 - adler, 3795
 - avail_in, 3795
 - avail_out, 3795
 - data_type, 3795
 - msg, 3795
 - next_in, 3795
 - next_out, 3795
 - opaque, 3795
 - reserved, 3795
 - state, 3795
 - total_in, 3795
 - total_out, 3795
 - zalloc, 3795
 - zfree, 3795
- z_streamp
 - zlib.h, 4215
- Z_SYNC_FLUSH
 - zlib.h, 4215
- Z_TEXT
 - zlib.h, 4215
- Z_TREES
 - zlib.h, 4215
- Z_UNKNOWN
 - zlib.h, 4215
- Z_VERSION_ERROR
 - zlib.h, 4215
- ZALLOC
 - zutil.h, 4217
- zalloc
 - z_stream_s, 3795
- zconf.h
 - Byte, 4211
 - Bytef, 4211
 - charf, 4211
 - const, 4211
 - FAR, 4211
 - intf, 4211
 - MAX_MEM_LEVEL, 4211
 - MAX_WBITS, 4211
 - OF, 4211
 - SEEK_CUR, 4211
 - SEEK_END, 4211
 - SEEK_SET, 4211
 - uInt, 4211
 - uIntf, 4211
 - uLong, 4211
 - uLongf, 4211
 - voidp, 4211
 - voidpc, 4211
 - voidpf, 4211
 - z_off64_t, 4211
 - z_off_t, 4211
 - ZEXPORT, 4211
 - ZEXPORTVA, 4211
 - ZEXTERN, 4211
- ZEXPORT
 - zconf.h, 4211
- ZEXPORTVA
 - zconf.h, 4211
- ZEXTERN
 - zconf.h, 4211
- ZFREE
 - zutil.h, 4217
- zfree
 - z_stream_s, 3795
- ZipException
 - decaf::util::zip::ZipException, 3796, 3797
- zlib.h
 - deflateInit, 4214
 - deflateInit2, 4214
 - gz_header, 4215
 - gz_headerp, 4215
 - gzFile, 4215
 - inflateBackInit, 4214
 - inflateInit, 4214
 - inflateInit2, 4215
 - OF, 4215
 - Z_ASCII, 4215
 - Z_BEST_COMPRESSION, 4215
 - Z_BEST_SPEED, 4215
 - Z_BINARY, 4215
 - Z_BLOCK, 4215
 - Z_BUF_ERROR, 4215
 - Z_DATA_ERROR, 4215
 - Z_DEFAULT_COMPRESSION, 4215
 - Z_DEFAULT_STRATEGY, 4215
 - Z_DEFLATED, 4215
 - Z_ERRNO, 4215
 - Z_FILTERED, 4215
 - Z_FINISH, 4215
 - Z_FIXED, 4215
 - Z_FULL_FLUSH, 4215
 - Z_HUFFMAN_ONLY, 4215
 - Z_MEM_ERROR, 4215
 - Z_NEED_DICT, 4215
 - Z_NO_COMPRESSION, 4215
 - Z_NO_FLUSH, 4215
 - Z_NULL, 4215
 - Z_OK, 4215
 - Z_PARTIAL_FLUSH, 4215
 - Z_RLE, 4215
 - z_stream, 4215

Z_STREAM_END, 4215
Z_STREAM_ERROR, 4215
z_stream, 4215
Z_SYNC_FLUSH, 4215
Z_TEXT, 4215
Z_TREES, 4215
Z_UNKNOWN, 4215
Z_VERSION_ERROR, 4215
ZLIB_VER_MAJOR, 4215
ZLIB_VER_MINOR, 4215
ZLIB_VER_REVISION, 4215
ZLIB_VER_SUBREVISION, 4215
ZLIB_VERNUM, 4215
ZLIB_VERSION, 4215
zlib_version, 4215
ZLIB_INTERNAL
 gzguts.h, 4204
 zutil.h, 4217
ZLIB_VER_MAJOR
 zlib.h, 4215
ZLIB_VER_MINOR
 zlib.h, 4215
ZLIB_VER_REVISION
 zlib.h, 4215
ZLIB_VER_SUBREVISION
 zlib.h, 4215
ZLIB_VERNUM
 zlib.h, 4215
ZLIB_VERSION
 zlib.h, 4215
zlib_version
 zlib.h, 4215
zstrerror
 gzguts.h, 4204
zutil.h
 Assert, 4217
 DEF_MEM_LEVEL, 4217
 DEF_WBITS, 4217
 DYN_TREES, 4217
 ERR_MSG, 4217
 ERR_RETURN, 4217
 F_OPEN, 4217
 local, 4217
 MAX_MATCH, 4217
 MIN_MATCH, 4217
 OF, 4217
 OS_CODE, 4217
 PRESET_DICT, 4217
 STATIC_TREES, 4217
 STORED_BLOCK, 4217
 Trace, 4217
 Tracec, 4217
 Tracecv, 4217
 Tracev, 4217
 Tracevv, 4217
 TRY_FREE, 4217
 uch, 4217
 uchf, 4217
 ulg, 4217
 ush, 4217
 ushf, 4217
 z_errmsg, 4217
 ZALLOC, 4217
 ZFREE, 4217
 ZLIB_INTERNAL, 4217